DYNAMIC MULTI-OBJECTIVE OPTIMIZATION: A TWO ARCHIVE STRATEGY

by

Renzhi Chen

A thesis submitted to
The University of Birmingham
for the degree of
DOCTOR OF PHILOSOPHY

School of Computer Science
College of Engineering and Physical Sciences
The University of Birmingham

ABSTRACT

Existing studies on dynamic multi-objective optimization mainly focus on dynamic problems with time-dependent objective functions. Few works have put efforts on dynamic problems with a changing number of objectives, or dynamic problems with time-dependent constraints. When problems have time-dependent objective functions, the shape or position of the Pareto-optimal front/set may change over time. However, when dealing with problems with a changing objective number or time-dependent constraints, the challenges are different. Changing number of objectives leads to the expansion or contraction of the dimensions of the Pareto-optimal front/set manifold, while time-dependent constraints may change the shape of feasible regions over time. The existing dynamic handling techniques can hardly handle the changing number of objectives. The state-of-arts in constraints handling techniques are incapable of tackling problems with time-dependent constraints. In this thesis, we present our attempts toward tackling 1) the dynamic multi-objective optimizing problems with a changing number of objectives and 2) multi-objective optimizing problems with time-dependent constraints. Two-archive Evolutionary Algorithms are proposed. Comprehensive experiments are conducted on various benchmark problems for both types of dynamics. Empirical results fully demonstrate the effectiveness of our proposed algorithms.

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to my supervisor Prof. Xin Yao, for his academic support in the past four years. His rich knowledge in the field of evolutionary algorithms and insightful comments in every meeting have deepened my understanding in this research area and helped me improve my research. I could not have imagined having a better supervisor than him for my PhD life.

I am also grateful to Dr. Ke Li. He is a very good mentor. He gave me many support and guidance on the projects in the past one year.

I thank all the members of my RSMG group: Prof. John Barnden, Prof. John Bullinaria, and Dr. Rami Bahsoon. They have provided helpful suggestions in my every RSMG report. Their good questions in the RSMG meetings inspires me to improve my research thereafter.

Last but not least, my deepest thanks go to my families. They have continuously given my mental support and encouragement whenever I encounter difficulties in life and work.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

Acronyms

MOP(s)	Multi-objective optimization problem(s)
DMO	Dynamic multi-objective optimization
DM(s)	Decision maker(s)
PS	Pareto-optimal set
PF	Pareto-optimal front
EA(s)	Evolutionary Algorithm(s)
MOEA(s)	Multi-objective evolutionary algorithm(s)
CA	Convergence archive
DA	Diversity archive
CMOEA(s)	Constrained multi-objective evolutionary algorithms
CMOP(s)	Constrained multi-objective optimization problem(s)
DMOPs	Dynamic multi-objective optimization problems
DCMOP(s)	Dynamic constrained multi-objective optimization problem(s)
RUP	Rational Unified Process
QoS	Quality of service
CV	Constraint violation
CDR	Constrained dominance relation
WDN	Water distribution network problems
IQR	interquartile range
GD	Generational Distance
IGD	Inverted Generational Distance
MIGD	Mean Inverted Generational Distance
HV	Hypervolume
MHV	Mean Hypervolume
LHS	Latin hypercube sampling

Algorithms

NSGA-II	Non-dominated sorting genetic algorithm II
SPEA2	Improved strength Pareto EA
IBEA	indicator-based evolutionary algorithm
MOEA/D	Multi-objective EA based on decomposition
TAEA	Two-archive evolutionary algorithms
DTAEA	Dynamic two-archive evolutionary algorithm
C-TAEA	Constrained two-archive EA
C-DTAEA	Constrained dynamic two-archive EA
DNSGA-II	Dynamic non-dominated sorting genetic algorithm II
MOEA/D-KF	Multi-objective EA based on decomposition with Kalman Filter predictor
C-MOEA/D	Constrained multi-objective EA based on decomposition
C-NSGA-III	Constrained non-dominated sorting genetic algorithm III
C-MOEA/DD	Constrained MOEA based on Pareto dominance and Decomposition
I-DBEA	Independent distances based EA
CMOEA	Algorithm for CMOPs proposed by Woldesenbet et al.
CMOEA/D	constrained multi-objective EA based on decomposition

CHAPTER 1

INTRODUCTION

Multi-objective optimization problems (MOPs) consider optimizing more than one conflicting objective simultaneously. MOPs arise in many engineering applications when optimal decisions need to be taken in the presence of two or more conflicting objectives. For example, in motor design [1], maximizing the engine performance and minimizing the pollution emission are two typical conflicting objectives in motor design. When optimizing the aerodynamics performance of a high-speed train head [2], minimizing the drag force and the lift force are conflicting with each other. In multi-objective optimization, there does not exist a global optimum, which optimizes all objectives. Instead, decision makers (DMs) are often interested in a set of trade-off solutions which scarifies one objective for the other. The concept of Pareto dominance is introduced to determine which solution is better in the comparison when considering multiple conflicting objectives. Specifically, solution A is said to dominate a solution B if and only if 1) solution A is not worse than solution B in all objectives and 2) solution A is better than solution B in at least one objective. A solution is called Pareto-optimal in case that there does not exist any other solution dominate it. The set of Pareto-optimal solutions in the decision space is called the Pareto-optimal set (PS), and its corresponding mapping in the objective space form the Pareto-optimal front (PF).

Evolutionary Algorithm (EA), inspired by biological evolution, has been recognized as a popular approach for multi-objective optimization. EAs maintain a set of solutions, also

known as a population, during the optimization process. In each iteration, offspring solutions are reproduced from current solutions through the crossover and mutation. These offspring solutions have possibilities to replace solutions in the current population during the selection process. EA has been widely accepted for solving MOPs because of its population-based nature which enables the approximation of a set of non-dominated solutions in a single simulation run. Generally speaking, there are three basic goals in evolutionary multi-objective optimization: 1) approximating the PF as close as possible (this is known as convergence), 2) making sure that the population distribution is as uniform as possible and 3) making sure that the population covers the whole PF as much as possible. In particular, the latter two are also known as diversity. In the past three decades and beyond, many efforts have been devoted to the development of multi-objective EAs for achieving the balance between convergence and diversity. Many multi-objective evolutionary algorithms (MOEAs) have been proposed over the past three decades, such as non-dominated sorting genetic algorithm II (NSGA-II) [3], indicator-based EA (IBEA) [4], multi-objective EA based on decomposition (MOEA/D) [5]. Praditwong and Yao proposed to use a two-archive mechanism [6, 7]. In particular, as convergence and diversity are two separate goals in EMO, their basic idea is to use two complementary populations to achieve the convergence and the diversity separately. As the first attempt, this two-archive mechanism has been proved to work on some benchmark problems and it also forms the algorithmic foundation of the methodologies developed in this thesis.

Furthermore, constrained optimization is ubiquitous in the real world, whereas studies on constrained multi-objective optimization (CMO) have been surprisingly lukewarm in the literature. Most, if not all, existing constraint handling techniques simply borrow ideas from single-objective optimization. In particular, they mainly aim at pushing the population toward the feasible region as much as possible before considering the balance between convergence and diversity within the feasible region. Furthermore, few works have pay attention to the importance of infeasible solutions. Most, if not all, existing constraint handling techniques tend to push a population toward the feasible region as

much as possible before considering the balance between convergence and diversity within the feasible region. Few works have pay attention to the value of infeasible solutions. However, in some complex situations, algorithms cannot handle the constrained problems without exploring the infeasible region or maintaining some infeasible solutions. One example can be found in the scenario where two feasible regions are separated by an infeasible barrier. It is hard to push a solution from one feasible region to the other without entering the infeasible barrier.

In addition, it is not uncommon that real-world optimization scenarios are fully of dynamics or uncertainties. In recent years, dynamic multi-objective optimization (DMO) has become increasingly popular in the EMO community. Generally speaking, a DMO problem (DMOP) can involve various types of dynamic characteristics, e.g. time-dependent objective and/or constraint functions, and a changing dimensionality (in terms of the number of variables or objectives). Thus, DMOP is intrinsically more challenging due to the existence of dynamics. Due to the population-based property and its self-adaptation, the EA has been widely used to solve DMOPs. However, the current studies on DMO mainly focus on problems whose function formulations will change over time, whereas very limited studies have been on considering the change of dimensionality (e.g. the number of objectives and/or constraints change over time). Note that this sort of dynamic problems are not uncommon in the real world where we will delineate in the later chapters. The challenges for MOPs with changing number of objectives are different from the MOPs with time-dependent objective functions. When objectives are added or removed during the search process, instead of changing the position or the shape of the original PS/PF, the dimension of PS/PF manifold will expand or contract. Besides the changing number of objectives, the constraints for MOPs may change during the optimization process as well. We name problems with dynamic constraints as dynamic constrained multi-objective optimization problems (DCMOPs). Dynamic constraints can be classified into many types. In this thesis, we only discuss problems with time-dependent constraints, which are not fully studied. Current dynamic handling techniques cannot handle this type of problems

as the challenges for DCMOPs come from two aspects: 1) finding well-converged and well-diversified feasible solutions before changes, and 2) tracking the moving feasible region after changes.

the following paragraphs, we first describe the motivations that lead to the research conducted in this thesis. Afterwards, we outline the main contributions of this thesis.

1.1 Motivations

EAs have been widely used for solving MOPs since the 1990s. Whereas many real-world scenarios consider problems in a uncertain environment, e.g. the objective and/or the constraint functions can change over time, these changes have an in-negligible impact on the process of optimization: the time-dependent objective functions may lead to a moving PS/PF; adding or removing objectives may cause the expanding or contracting of the dimension of PS/PF manifold; changing constraints may cause a changing feasible region. These challenges bring about the emerging interests for the research for DMO. In the past decade, many efforts have been devoted to develop various methods for handling problems with a time-dependent objective functions [8, 9]. Whereas, another dynamic scenario, i.e. problems with a changing number of objectives which is not uncommon in the real-world scenarios, has rarely been studied in the literature. In particular, we will describe some scenarios that lead to such dynamic characteristics.

- The number of objective functions changes naturally as a requirement from the real-life optimization scenarios. For example, in the model of Rational Unified Process (RUP) in software project scheduling problem, the number of quality function changes as the project steps into different phases, resulting in a changing number of objectives. Similar situations also happen in scheduling resources under a cloud computing environment, where the quality of service (QoS) for a particular user is an objective in the optimization process. The number objectives for QoS varies when the number of users changes.

- The number of objective functions possibly changed as the expectation of DMs. In business management, there exist many operational aspirations/objectives. The performance objectives of operations management include quality, speed, flexibility, dependability and cost [10]. The optimization process tries to find trade-offs among these objectives. As a DM, (s)he can choose to optimize different objectives given the requirements at a particular operational stage.

- Modelling real-world optimization scenarios is very challenging in some cases. It is difficult to request a thoroughly considered optimization problem before the optimization process starts. Allowing the model builder to revise the model and change the number of objectives during the optimization can reduce the difficulty of modelling. The model builder can bear less pressure in the modelling as they still have chances to change afterwards during the optimization process.

Different from the challenges from the time-dependent objective functions, the DMOPs with a changing number of objectives usually cause the expansion or contraction of the dimension of the PS/PF manifold. Generally speaking, there are two key characteristics of DMOPs with a changing number of objectives. First, the Pareto-optimal solutions before the change will still be Pareto optimal after the increase of the number of objectives. Second, as the dimension of the PS/PF manifold will be changed after changing the number of objectives, the Pareto-optimal solutions before the change will be either crowded in a small area on the PF after the change, or partly drifted away from the PF after the change. These characteristics will be further discussed in the Section 3.1. Algorithms designed for the DMOPs with time-dependent objective functions have difficulties to handle this problem. Prediction-based methods such as PPS [8] aim to track the centre of the PF. By relocating solutions according to the predicted centre, these methods can successfully track the moving of PF. However, when the dimension of PF manifold expands or contracts, rather than moving, these relocating methods fail. Introducing diversified solutions such as DNSGA-II [9] also cannot tackle the challenges. The reasons are as follow: 1) for the situation of increasing number of objectives, a few newly

introduced solutions are easily dominated by current solutions, as current solutions are all close to the PF. Adding more solutions can solve the problem. However, it will also ruin the convergence of the population, as too many random solutions are in the population. 2) For the situation of decreasing number of objectives, the challenge for this situation is that many solutions are drifted away from the PF, whereas the newly introduced solutions cannot help push the drifted solutions back to the PF. The historical information only works well on periodic changes.

Besides of the DMOPs, many techniques have been proposed to handle the CMOPs. One of the major issues of solving CMOPs is how to deal with the infeasible solutions throughout the search process. Algorithms designed for CMOPs have constraints handling techniques, which push solutions from infeasible region to the feasible region. Most of the constraints handling techniques rely on an certain "indicator" which measures the "quality" of a solution. For example 1) feasible solutions are better than infeasible ones, 2) solutions closer to the feasible region are preferable than those far apart from the feasible boundary. It is worth to mention that constraint violation (CV) is one of the most popular indicators developed along this line. Many MOEA can successfully handle CMOPs when using the CV based constrained dominance relation (CDR), such as CNSGA-II, CNSGA-III and CMOEA/D. However, these algorithms overemphasize the importance of feasibility. This can lead to an ineffective search when the algorithms encounter complex constraints. First, the optimization process can be misled by the local optima of a particular indicator. For example, a solution with a larger CV can be much closer to the feasible region than a solution with a smaller CV in a complex situation. Second, algorithms may fail to locate all feasible regions in case they are disparately distributed in the search space. Take a situation of two separated feasible regions as an example. When one feasible region is explored first, solutions will quickly gather in this region as algorithms prefer feasible solutions than infeasible ones. However, it is very difficult for solutions to go through the infeasible region from this feasible region to the other feasible region.

It is inarguable that problems become even more challenging when having a time-varying constraint. To the best of my knowledge, no algorithm has been developed for DCMOPs. Handling DCMOPs with algorithms designed for CMOPs has drawbacks. First, newly generated feasible solution will replace infeasible solution which closes to the PF but becomes infeasible after changes. This may ruins the convergence of the population, especially in a frequently changing situation. Second, the changing constraints may make the constraints handling techniques fail. The algorithms may always track the moving local optimal for a particular indicator, e.g. CV, rather than push solutions towards the PF. It is even more challenging to combine all the dynamics mentioned above. In the situation of DMOPs with both a changing number of objectives and dynamic constraints, there are much more issues need to be considered.

Overall, the DMOPs with a changing number of objectives are not uncommon, whereas the current algorithms designed for the DMOPs with time-dependent objective functions cannot handle this. Besides, there are some drawbacks for the existing algorithms for CMOPs. In addition, no research have been done for MOPs with dynamic constraints by the researchers. These problems challenges the current algorithms for static constraints. The DMOP with a changing number of objectives together with dynamic constraints is a more complicated situation to deal with. All these suggest that we develop algorithms to tackle these problems.

1.2 Contributions

The main contributions of the thesis are outlined as follows:

- We have conducted an analysis and discussion on the challenges of DMOPs with a changing number of objectives in Section 3.1. The main challenges for DMOPs with a changing number of objectives are: 1) increasing number of objectives impacts slightly for the convergence of the population, but it ruins the diversity of the population. 2) decreasing number of objectives drags some solutions into the

search space, but the rest of the solutions stay close to the PF. We have significantly enhanced the two-archive evolutionary algorithm, by adapting to the DMOPs with a changing number of objectives. The algorithm, named dynamic two-archive evolutionary algorithm (DTAEA), are designed according to the characteristics for the DMOPs with a changing number of objectives. Similar to other two-archive algorithms, the CA in DTAEA prefer solutions close to the PF, while the diversity archive maintains solutions which can enhance the diversity. Unlike other two-archive algorithms, DTAEA stresses that the DA is a complement to the CA. Based on this and the characteristics of DMOPs with a changing number of objectives, the update, selection and reproduction mechanisms are designed. The two main characteristics are: 1) the PS/PF before the change is a subset of the one after increasing the number of objectives, and vice versa; and 2) instead of changing the position or the shape of the original PS/PF, increasing or decreasing the number of objectives usually results in the expansion and contraction of the dimension of the PS/PF manifold. In the experiments, DTAEA shows better results in most of the cases compared with other four state-of-art algorithms designed for DMOPs or MOPs. Moreover, several research questions have been raised and discussed, which include further understanding of how the component contributes to DTAEA, in what conditions DTAEA may fail. In addition, a new approach solving the many-objective optimization problem is proposed.

- We have proposed a two-archive EA, denoted as C-TAEA, to solve constrained MOPs. We discuss the challenges for the existing algorithms to solve constrained MOPs. Although current algorithms for constraints handling perform well in many cases, they cannot handle complicated searching situations. Based on the same idea of DTAEA, the main characteristics of C-TAEA are 1) the CA prefers well-converged and feasible solutions, which provides selection pressures toward the PF. 2)the DA explores the areas that have not been or will not be exploited by the CA, including areas in which CA cannot find well-converged solutions in infeasible

areas. Due to the lack of test problems for scalable constrained MOPs, we proposed our own test problems, which is denoted as DC-DTLZ series. In the experiments, C-TAEA shows better results in many cases, including the test problems of the constrained DTLZ (C-DTLZ) series [11] and self-designed DC-DTLZ series. Then we adapt the C-TAEA for the MOPs with dynamic constraints. By adding the reconstructing mechanism and adapting the mating mechanism for the dynamic situation, the C-TAEA can handle the dynamic constraints. We test the algorithm on the test problems which adapted from C-DTLZ and DC-DTLZ series. The dynamic constraints includes only time-dependent constraints functions. A changing number of constraints is not considered. The C-TAEA performs well on all the situations while other algorithms fail.

- We have adapted the C-TAEA to the water distribution network problems (WDN). The WDN is a real-life case study with four objectives and two time-dependent constraints. We use C-TAEA to solve one of the most difficult test problems for WDN, known as anytown WDN. The performance of C-TAEA is better than other five algorithms.

- We have enhanced and combined the C-TAEA and DTAEA for the DMOPs with a changing number of objectives and dynamic constraints. We have conducted a analysis and discussion on the DMOPs with a changing number of objectives. The algorithm, named constrained dynamic two-archive evolutionary algorithm (C-DTAEA), is proposed. C-DTAEA is mainly designed to handle a changing number of objectives as well as dynamic constraints. Although these scenarios are different, the basic requirement for the algorithms is: the algorithms should have strong ability to explore the search space. While the CA behaves similar to all other algorithms, the DA in C-DTAEA is the key role. The DA will maintain non-dominated but infeasible solutions, as well as dominated solution, but helpful to the diversity. On the proposed test problems as well as the the dynamic C-DTLZ series adapted from

C-DTLZ series [11], C-DTAEA achieves a excellent results: C-DTAEA is better than other five algorithms on 15 out of 16 case.

1.3 Publications

- Renzhi Chen, Peter R. Lewis, Xin Yao: Temperature management for heterogeneous multi-core FPGAs using adaptive evolutionary multi-objective approaches. ICES 2014: 101-108

- Renzhi Chen, Ke Li, Xin Yao: Dynamic Multi-objectives Optimization With a Changing Number of Objectives. IEEE Trans. Evolutionary Computation 22(1): 157-171 (2018)

- Ke Li, Renzhi Chen, Guangtao Fu and Xin Yao: Two-Archive Evolutionary Algorithm for Constrained Multi-Objective Optimization. IEEE Trans. on Evolutionary Computation: online on 19/7/2018.

- Ke Li, Renzhi Chen, Geyong Min and Xin Yao: Integration of Preferences in Decomposition Multi-Objective Optimization. IEEE Trans. on Cybernetics: published online on 20/8/2018.

CHAPTER 2

BACKGROUND

This chapter will at first provide some basic knowledge useful in this thesis. Afterwards, a literature review will lead you to overview some important developments in dynamic multi-objective optimization and constrained multi-objective optimization. This chapter is organised as follow. Section 2.1 describes the basic concepts for multi-objective optimisation and dynamic multi-objective optimisation. Section 2.2 introduces the evolutionary multi-objective optimization. Section 2.3 provides a review of evolutionary constrained multi-objective optimization. A brief introduction for evolutionary dynamic multi-objective optimization is given in Section 2.4.

2.1 Basic Concepts

This section includes the basic concepts for MOPs (Section 2.1.1), and DMOPs (Section 2.1.2).

2.1.1 Multi-objective Optimization

The multi-objective optimization problem (MOP) considered in this thesis is mathematically defined as:

$$\begin{aligned} \text{minimize} \quad & \mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), \cdots, f_m(\mathbf{x}))^T \\ \text{subject to} \quad & \mathbf{G}(\mathbf{x}) \leq 0 \\ & \mathbf{H}(\mathbf{x}) = 0 \\ & \mathbf{x} \in \mathbf{\Omega} \end{aligned} \tag{2.1}$$

where $\mathbf{x}$ is a vector of n decision variables: $\mathbf{x} = (x_1, x_2, ..., x_n)^T, \mathbf{x} \in \mathbf{\Omega}$. $\mathbf{\Omega} \subseteq \mathbb{R}^n$ is the decision (variable) space.

The constrained functions include p inequality constraints $\mathbf{G}(\mathbf{x}) = (G_1(\mathbf{x}), G_2(\mathbf{x}), ..., G_p(\mathbf{x}))^T$ and q equality constraints $\mathbf{H}(\mathbf{x}) = (H_1(\mathbf{x}), H_2(\mathbf{x}), ..., H_q(\mathbf{x}))^T$. Solution $\mathbf{x}$ is called *feasible solution* if it satisfies all the constraints functions. A solution violating any constraints is called *infeasible solution*.

Involving more than one objective function to be optimized simultaneously, most often, no single solution exists, which optimizes every objective for a non-trivial multi-objective optimization problem. Instead of a single solution, a set of solutions, denoted as Pareto-optimal solutions, is used to reflect the preference of the problem. Pareto dominance is used to define this preference. A solution $\mathbf{x}^1 \in \Omega$ is said to be *Pareto dominate* solution $\mathbf{x}^2 \in \Omega$, denoted by $\mathbf{x}^1 \prec \mathbf{x}^2$, if and only if the following condition is satisfied:

$$\forall i \in \{1, 2, ..., m\}, f_i(\mathbf{x}^1) \leq f_i(\mathbf{x}^2) \ \wedge \ \exists j \in \{1, 2, ..., m\}, f_j(\mathbf{x}^1) < f_j(\mathbf{x}^2)$$

A solution $\mathbf{x}$ is said to be *Pareto optimal*, if and only if $\nexists \mathbf{x}^* \in \Omega, \mathbf{x}^* \prec \mathbf{x}$. The set of Pareto-optimal solutions in the decision space is called as Pareto-optimal Set (PS), $PS = \{\mathbf{x} \in \Omega | \nexists \mathbf{x}^* \in \Omega, \mathbf{x}^* \prec \mathbf{x}\}$. The corresponding set of vectors in the objective space forms the PF, $PF = \{\mathbf{F}(\mathbf{x}) | \mathbf{x} \in PS\}$.

2.1.2 Dynamic Multi-objective Optimization

There are various types of dynamic characteristics that can result in different mathematical definitions. Farina et al. [12] only includes problems with changing objective functions in their definition. Raquel et al. [13] adapt the definition by Nguyen et al. [14], which includes time-linkage dynamic problems. Huang et al. [15] propose their definition which includes both changing the number of objective functions and changing the number of constraints. The focus of this thesis mainly stays on the continuous DMOPs defined as follows:

$$\begin{aligned} \text{minimize} \quad & \mathbf{F}(\mathbf{x}, t) = (f_1(\mathbf{x}, t), \cdots, f_{m(t)}(\mathbf{x}, t))^T \\ \text{subject to} \quad & \mathbf{G}(\mathbf{x}, t) = (G_1(\mathbf{x}, t), G_2(\mathbf{x}, t), ..., G_{p(t)}(\mathbf{x}), t)^T \leq 0 \\ & \mathbf{H}(\mathbf{x}, t) = (H_1(\mathbf{x}, t), H_2(\mathbf{x}, t), ..., H_{q(t)}(\mathbf{x}, t))^T = 0 \\ & \mathbf{x} \in \mathbf{\Omega}_t \\ & t \in \mathbf{T} \end{aligned} \tag{2.2}$$

$\mathbf{T} = \{t_1, t_2, \cdots, t_{max}\} \subseteq \mathbb{N}$ is the set of time steps of the optimization process. t_i is the i-th time step in the optimization process. $\Omega_t \subseteq \mathbb{R}^{n(t)}$ is the decision (variable) space at time step t, $\mathbf{x} = (x_1, \cdots, x_{n(t)})^T \in \Omega_t$ is a candidate solution. $\mathbf{F} : \Omega \to \mathbb{R}^{m(t)}$ constitutes of $m(t)$ real-valued objective functions and $\mathbb{R}^{m(t)}$ is called the objective space at time step t. The constrained functions includes $p(t)$ inequality constraints $\mathbf{G}(\mathbf{x}, t)$ and $q(t)$ equality constraints $\mathbf{H}(\mathbf{x}, t)$. $n(t)$, $m(t)$, $p(t)$ and $q(t)$ are functions of t.

At time step t, $\mathbf{x}^1$ is said to *Pareto dominate* $\mathbf{x}^2$, denoted as $\mathbf{x}^1 \prec_t \mathbf{x}^2$, if and only if the following condition is satisfied:

$$\forall i \in \{1, \cdots, m(t)\}, f_i(\mathbf{x}^1, t) \leq f_i(\mathbf{x}^2, t) \wedge \exists j \in \{1, \cdots, m(t)\}, f_j(\mathbf{x}^1, t) < f_j(\mathbf{x}^2, t)$$

At time step t, $\mathbf{x} \in \Omega$ is said to be *Pareto-optimal*, if and only if $\nexists \mathbf{x}^* \in \Omega, \mathbf{x}^* \prec_t \mathbf{x}$. The set of all Pareto-optimal solutions is called the t-th PS (denoted as PS_t), $PS_t = \{\mathbf{x} \in \Omega | \nexists \mathbf{x}^* \in \Omega, \mathbf{x}^* \prec_t \mathbf{x}\}$. The corresponding set of Pareto-optimal objective vectors is called

the t-th PF (denoted as PF_t), $PF_t = \{\mathbf{F}(\mathbf{x}, t) | \mathbf{x} \in PS_t\}$.

2.2 Basic knowledge of Evolutionary Multi-objective Optimization

This section provides some basic knowledge related to the use of EAs for solving MOPs, as known as evolutionary multi-objective optimization. It includes an introduction of basic elements of an MOEA, some existing develops in evolutionary multi-objective optimization, indicators for performance assessment and benchmark problems.

2.2.1 Introduction

By mimicking the process of natural evolution, EAs become one of the most important optimization methods. EAs are found to be efficient when dealing with MOPs. EAs have been used to solve MOPs in many areas including chemistry [16], physic [17], engineering [18] and etc. This is because of its population-based characteristic which enables EA to approximate a set of trade-off alternatives in a single run. In multi-objective optimization, achieving a balance between convergence and diversity is essential, i.e. finding a set of solutions not only approximate the PF as close as possible (as known as convergence) but also have a well distribution (as known as diversity).

Generally speaking, a basic MOEA includes the following major steps.

- Step 1 Generate the initial population of solutions. (denoted as population initialization)
- Step 2 Evaluate the fitness of each solutions in the population.
- Step 3 Repeat the following regeneration steps until termination:
  - Step 3.1 Select solutions for reproduction. (denoted as mating selection)
  - Step 3.2 Generate offspring through crossover and mutation.

- Step 3.3 Evaluate the fitness of each solutions in the offspring.
- Step 3.4 Combine the parent and the offspring populations; and select the fittest solutions to form the next parents. (denoted as environmental selection)

In the following paragraphs, I will further explain some more details of the above mentioned steps.

Population Initialization

Population initialization generates solutions as the initial population for EAs, which is an important part of EAs. The initial population of a evolutionary algorithm can affect the convergence speed and also the quality of the final result [19]. Without any prior knowledge from DMs, random sampling is the most common method to generate candidate solutions. Here, some popular sampling methods are outlined as follows.

- The default uniform pseudo-random number generator are commonly applied in most of the algorithms. This generator is usually applied in the complier's standard library, such as the `rand()` in the C library and RANDOM class in the Java library.
- *Latin Hypercube Sampling (LHS)* [20] can reduce the variance in the Monte Carlo estimate of the integrand. The range of the variables is divided into intervals of equal probability. The variables are selected randomly according to the probability density in the interval. Then random pairing based on a pseudo-random number generator for all variables are employed to formulate the final samples. LHS is applied in András Szöllös' work [21] and the algorithms proposed in this thesis [22].
- *Cross Validation* [23] estimates the error of the model and chooses a new set of points to make the response surface more reliable. *Cross Validation* is efficient when dealing with the ZDT functions [24] in Silvia's work [19].
- *Hammersley sequence sampling technique* [25] is used to generate the initial population by Jamali et al. and Martinez et al. [26, 27]. This technique is another

quasi-random number technique which uses the Hammersley points to uniformly sample a (k1)-dimensional hypercube.

Mating Selection

Mating selection aims at choosing promising individuals from the current population for reproduction. Solutions better than others are naturally more likely to reproduce their genetic information. Interestingly, mating strategies in EAs do not attract much attentions. The mating selection is usually performed in a random way. Nevertheless, there do exist some studies designing distinct strategies for mating selection.

- *Tournament selection* [28] is the most welcomed method of selecting a solution from a population in genetic algorithms. Tournament selection runs several "tournaments" among a few solutions chosen randomly from the union of population and offspring. The one with the best fitness is selected.

- *Restricted mating strategy‘* selects solutions randomly. However, the mating is permitted only when the distance between two solutions is large enough. This method may avoid the formation of lethal solutions and therefore improve the online performance [29].

- *Mating pool strategy* maintains a subset of the union of current population and the offspring. Only members of this subset participate in the mating selection process. The strength Pareto evolutionary algorithm II (SPEA2) [30] is the most well-known algorithm used this strategy.

Environmental Selection

Environmental selection is one of the most critical steps in MOEAs. Some solutions in the offspring are selected to replace the solutions in the current population. The basic principle for environmental selection is that solutions better than others are more likely

to survive. Most of the algorithms select solutions based on the Pareto dominance relation and density. The non-dominated solutions are more likely to be selected. Since the dominance relation does not reflect the diversity information of the population, density information usually is integrated into the selection process: the lower density of the surrounding area of a solution, the higher chance of this solution survived in the environmental selection. Besides the algorithms based on the Pareto dominance relation and diversity maintenance, there are other environmental selection strategies, such as integrating the Pareto dominance relation and diversity information into one indicator, or decomposing the multi-objective problem into a series of scalar optimisation problems.

2.2.2 Methods

EAs for MOPs can be classified into three groups: Pareto-based algorithms, decomposition-based algorithms and indicator-based algorithms.

Pareto-based algorithms

The basic idea for algorithms in this group is sorting and selecting the solutions according to the Pareto dominance relation and density. Algorithms usually apply different diversity maintenance strategies. Niching technique is first used by Holland et al. [31]. The crowding distance in non-dominated sorting genetic algorithm II (NSGA-II) [32] and adapting scatter search (AbYSS) [33] is one of the most well-known and efficient strategies dealing multi-objectives problems. Other strategies such as $k-$th nearest neighbour [30, 34] and grid crowding degree [35]. According to the literature, the Pareto-based algorithms perform well in multi-objective problems. However, the Pareto dominance relation becomes less effective when during many objective problems. It is hard for algorithms in this group to handle many-objective optimization problems.

Decomposition-based algorithms

Unlike the Pareto-based algorithms, algorithms in this group guarantee the density by decomposing an MOP into a number of scalar objective optimization problems (sub-problems). A set of well-distributed reference points guide the solutions towards the optimum for every sub-problems. The most well-known algorithm in this group is the multi-objective evolutionary algorithm based on decomposition (MOEA/D) [36]. In MOEA/D, each sub-problem keeps one solution. For each sub-problem, algorithm generates a new solution by performing evolutionary process on several solutions kept in the neighbouring sub-problems. The sub-problem will update its solutions if the newly generated solution is better. There have been a number of improvements on MOEA/Ds proposed recently. Studies have been carried out about adaptively allocating the computational resources to different sub-problems [37–40], reference points generation methods [37, 41, 42], modifying the reproduction operators [43–45] and replacement strategies [46, 47]. Non-dominated sorting genetic algorithm III (NSGA-III) [48] is one of the state-of-art algorithms in this group. Unlike most of the other decomposition-based algorithms, NSGA-III do not tie a solution to every sub-problem. By contrast, solutions are associated with one sub-problem in NSGA-III, which means multiple or zero solutions may be tied to a sub-problem.

Indicator-based algorithms

The indicator-based algorithms use a performance indicator to guide the searching process. The basic idea is different from the Pareto-based algorithms: instead of using Pareto dominance relation and density separately, the algorithms use a single fitness function, combining Pareto dominance relation and density together, guiding the searching process. The indicator-based evolutionary algorithm (IBEA) [49] is the first work proposed in this group. Other indicator-based algorithms include SMS-EMOA [50] and MO-CMA-ES [51]. The main drawback of indicator-based algorithms is the computational time for calculating the indicator values.

2.2.3 Performance Indicators

As its name suggests, the performance indicator is used to measure the quality of approximated solutions obtained by MOEAs. In the past decades, a variety of performance indicators have been proposed. The indicators mainly measure three aspects: 1) whether the solutions are close to the PF, 2) whether the solutions cover the whole PF, 3) whether the solutions are well spread. The first aspect is also known as convergence requirement, while the second and third aspects usually are considered together as the diversity requirements. Some indicators only involve either convergence or diversity. Generational Distance(GD) [52], Convergence Measure [53], Error Ratio [54], Dominance Ranking [55] and Purity [56] only involve the convergence, while Spacing [57], Uniform Distribution [58], Δ-Metric [32] and σ-Diversity Measure [59] only involve diversity. Some other indicators consider both diversity and convergence, the most well-known indicators include Hypervolume (HV) [60], Inverted Generational Distance (IGD) [61], ϵ-Indicator [62] and G-Metric [63]. We briefly introduce HV and IGD which are used in the following chapters.

The IGD indicator calculates the average distance from the uniform distributed points in the true Pareto front to their nearest solutions. P^* is a set of points uniformly sampled along the true Pareto front, $\mathbf{S}$ is the set of solutions obtained by an EMO algorithm,

$$\mathrm{IGD}(P^*, \mathbf{S}) = \frac{\sum_{\mathbf{x} \in P^*} dist(\mathbf{x}, \mathbf{S})}{|\mathbf{S}|}, \tag{2.3}$$

where $dist(\mathbf{x}, P^*)$ is the Euclidean distance between a point P^* and its nearest solution in $\mathbf{x} \in S$. Note that the calculation of IGD requires the prior knowledge of the PF.

The hypervolume (HV) indicator calculates the volume of the objective space enclosed by the solutions and a reference point $\mathbf{z}$.

$$\mathrm{HV}(S) = \mathrm{VOL}(\bigcup_{\mathbf{x} \in S} [f_1(\mathbf{x}), z_1^w] \times \ldots [f_m(\mathbf{x}), z_m^w]) \tag{2.4}$$

where $\mathrm{VOL}(\cdot)$ indicates the Lebesgue measure. Note that solutions, dominated by the

worst point, are discarded for HV calculation.

2.2.4 Test Problems

Many test problems have been proposed to be the benchmarks for multi-objective optimization problems. The most widely used continuous test problems includes ZDT [24], DTLZ [64] and WFG [65].

The ZDT test suite proposed by Zitzler et al. includes six test problems. ZDT1 and ZDT4 have a convex Pareto front, and ZDT2 and ZDT6 have a concave Pareto front. Noted that due to being binary encoded, ZDT5 has often been omitted from analysis in related research. ZDT3 has a disconnected front. ZDT4 and ZDT6 are multi-modal problems. Moreover, ZDT6 is a many-to-one and biased (non-uniform mapping) problem. ZDT test suite has some advantages. First, the Pareto front of all the test problems in ZDT test suite are well defined. Second, ZDT test suite is widely employed in EA literature, which facilitates comparisons with other algorithms. However, the disadvantage for ZDT are obvious: problems in ZDT test suite only have two objectives.

The DTLZ test suite proposed by Deb et al. [64] is perhaps the most welcomed benchmarks in recent research. DTLZ test suite includes seven test problems: DTLZ1 is a biased problem with a linear Pareto front, DTLZ2 is an easy test problem with a spherical Pareto front, DTLZ3 is a multi-modal problem and DTLZ4 is a biased problem. DTLZ3 and DTLZ4 have the same Pareto front. DTLZ5 and DTLZ6 are degenerated problem. DTLZ7 has a disconnected Pareto front. DTLZ test suite introduces the idea of scalability, which means the number of objectives are scalable to any number of objectives. This is an important feature, which facilitated algorithms investigating into many objective problems.

The WFG suite includes nine test problems, which are also scalable in number of objectives and decision variables as DTLZ test suite. Compared with other test suites, WFG suite is more challenging. The WFG suite introduces attributes such as separability/non-separability, uni-modality/multi-modality and unbiased/biased parameters. The biggest

disadvantage for WFG test suite is that some of the Pareto front in the suite is too hard to achieve, due to the fact that most of the problems in WFG are non-separable.

2.3 Basic knowledge of Evolutionary Constrained Multi-objective Optimization

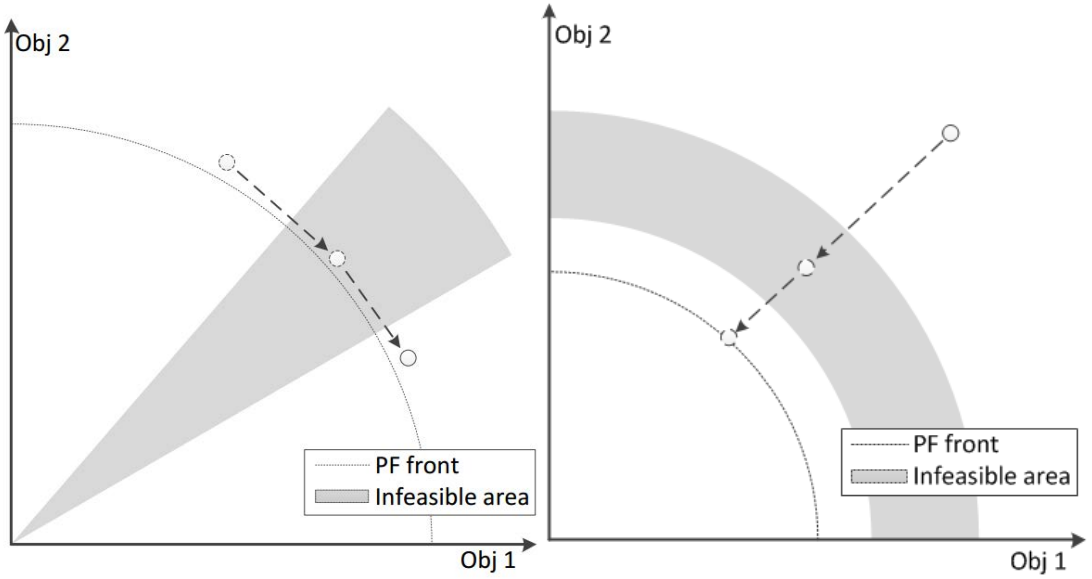
This section introduces the evolutionary approach for constraint multi-objective optimization, with the discussion of the basic concepts, the existing works and test problems.

2.3.1 Introduction

Algorithms need to optimize more than one objective simultaneously subject to the constraints in CMOPs. In MOPs, algorithms need to achieve a trade-off between convergence and diversity. As mentioned in Section 2.2 convergence represents whether solutions are close to the PF, and diversity represents whether solutions are well spread and covered the whole PF. Unlike MOPs, one more issue, denoted as feasibility, need to be balanced. The feasibility represents whether algorithms can find feasible solutions.

Feasibility maintenance is one of the major tasks for algorithms to handle CMOPs. However, few algorithms have noticed that in some situations, the infeasible solutions are also important to the optimization process. Although solutions located in the infeasible region are not counted as valid output, the infeasible solutions may facilitate the searching process. Figure 2.1 shows two cases. In Case-A, shown in Figure 2.1a, the infeasible region separate the search space into two feasible regions. The infeasible solutions may help algorithms to pass through the infeasible barrier between two feasible regions. In Case-B in Figure 2.1b, there is a infeasible ring between two feasible regions. Without maintaining infeasible solutions in the population, it is hard to converge to the PF.

Generally speaking, it is more challenging to handle CMOPs than MOPs, because algorithms need to achieve the balance among convergence, diversity and feasibility. Constrained multi-objective evolutionary algorithms (CMOEAs) are particularly designed to



(a) Case A (b) Case B

Figure 2.1: Importance of feasibility maintenance

solve CMOPs, with the capability of balancing the search between the feasible and infeasible areas in the search space [66]. The idea of constraint handling technique is extended from works for the constrained single objective optimization. Unlike constrained single objective optimization, only a few work has been done on solving constrained multi-objective optimization problems. When dealing with CMOPs, the target is to balance convergence, diversity and feasibility simultaneously. Algorithms usually use the constraint violation (CV) to indicate the feasibility of a solution. The CV can be defined in many ways. Usually a solution closer to the feasible region has a smaller CV. One of the commonly used definitions of CV is:

$$CV(\mathbf{x}) = \sum_{j=1}^{p+q} c_j(\mathbf{x}) \tag{2.5}$$

c_j is the degree of j-th constraint:

$$c_j = \begin{cases} |\min(G_j(\mathbf{x}), 0)| & j \leq p \\ |H_{j-p}(\mathbf{x})| & j > p \end{cases} \tag{2.6}$$

If $CV(\mathbf{x}) = 0$, $\mathbf{x}$ is feasible solution, otherwise $\mathbf{x}$ is an infeasible solution.

No performance indicators are developed for CMOPs in the previous works. In the literature, indicators developed for MOPs are used by ignoring all infeasible solutions.

2.3.2 Methods

Generally speaking, the existing constraint handling techniques for CMOPs can be divided into three groups.

Feasibility-driven algorithms

In this group, feasible solutions are granted a higher priority to survive than infeasible ones. Fonseca and Flemming [67] have developed a framework for solving CMOPs by assigning a higher priority to constraints than to objective functions. The algorithm proposed by Coello Coello and Christiansen simply discards all infeasible solutions [68]. Deb et al. have developed the constrained dominance relation (CDR) for the CMOPs [32]. In CDR, a solution x^1 dominates x^2 if 1) x^1 and x^2 are feasible and x^1 Pareto dominate x^2; 2) x^1 is feasible while x^2 is infeasible; 3)x^1 and x^2 are infeasible, but $CV(x^1) < CV(x^2)$. The CDR can simply cooperate with the Pareto dominance relation, which enables all Pareto-based algorithms to handle constraints. Algorithms developed based on CDR such as C-NSGA-II [11] and C-NSGA-III [48] perform well when dealing with constraints. Based on the same principle of CDR, the decomposition-based algorithms also can handle constrained problems using the CV as a prior criterion in the sub-problem update procedure. Several MOEA/D variants [11, 69, 70] have been developed under this idea. Different from CDR [32], Oyama et al. [71] have modified the CDR so that an algorithm prefers solutions violating fewer number of constraints. Takahama et al. [72] and Martnez et al. [73] have proposed an ϵ-CDR in which two solutions are considered as equal when the difference of their CVs is smaller than a threshold ϵ. Asafuddoula et al. [74] have proposed an adaptive ϵ-CDR where the threshold is dynamically adjusted according to the number of feasible solutions.

Convergence/feasibility balancing algorithms

In this group, algorithms try to achieve a trade-off between convergence and the feasibility. Li et al. [75] have developed a method that keeps solutions in the isolated regions even if they are infeasible. Ning et al. [76] have proposed a constrained non-dominated sorting method, which conducts a second non-dominated sorting with non-domination rank and constraint rank as two objectives. Woldesenbet et al. [77] have extended the idea of penalty function from technique for constrained single objective optimization problems into CMOPs, in which penalties are added to all the original objectives functions.

Feasible maintenance algorithms

In the last group, the algorithms try to avoid infeasible solutions in the reproduction step. Any infeasible solution is driven towards feasible region after generated. Harada et al. [78] have proposed a Pareto descent repair operator that explores possible feasible solutions around infeasible solutions in the constraint space. Jiao et al. Jiao et al. [79] have developed a feasible-guiding strategy in which a infeasible solution is guided moving towards its closest feasible solutions.

2.3.3 Test Problems

Only a few test problems are developed as the test bench for CMOPs. BNH [80], OSY [81], TNK [82], SRN [83], CTP [84] test suite, CF [85] test suite and the C-DTLZ test suite are widely used in the literature.

BNH [80], OSY [81], TNK [82] and SRN [83] are constrained two-objective problems with different types of constraints. BNH test problem has two nonlinear objective functions and two nonlinear inequality constraints. OSY test problem has two nonlinear objective functions, four linear inequality constraints and two nonlinear inequality constraints. TNK Test problem has two nonlinear objective functions, two linear inequality constraints. SRN test problem has two nonlinear objective functions, one linear inequality

constraints and a linear inequality constraint.

CTP test problems are have the capability of adjusting the difficulty of the constraint functions. The shape for the PF is not impacted by the constraints in the CTP test problems. The test problem CTP1 gives the difficulty near the PF, because the search area near the PF is infeasible. The constraints for test problems CTP2-CTP8 cause difficulty for convergence in the whole search space.

CF test problems are also commonly used benchmarks. For CF1-CF3 and CF8-CF10, the constraints do not impact the shape of the PF. The constraints only cause difficulty for convergence. The constraints for CF4-CF7 shaped their PFs into many optimal points which lie on some boundaries of the constraints.

2.4 Basic knowledge of Evolutionary Dynamic Multi-objective Optimization

This section introduces the dynamic multi-objective optimization. The first part gives the introduction. The second part of this section introduces the existing methods solving the DMOPs.

2.4.1 Introduction

Dynamic multi-objective optimization problems (DMOPs) are formally defined by Farina [12] in 2004. Compared with other optimization problems, DMOPs is quite new and haven't been fully studied. Following the definition in Equation 2.2, types of dynamic are listed as follows:

- Time-dependent constraints. Noted that the changes of constraints can be both changing number of constraints and changing constrained functions. The impact of these two types of changes are the same. The time-dependent decision space also counts in this type, as it is the change of boundaries constraints. It is a very

interesting research area, however, no existing works have been put efforts on this yet.

- Time-dependent objective functions. The situation of objective functions $\mathbf{F}$ varies during the optimization process is the most commonly discussed class of DMOPs. This type of changes are welled studied in the literature, which are commonly found in real world cases. According to Farina et al. [12], the time-dependent objective functions cause four types of impact:
  - TYPE I: Moving PF and PS.
  - TYPE II: Moving PF, static PS.
  - TYPE III: Moving PS, static PF.
  - TYPE IIII: Static PF and PS.
- Time-dependent dimension of decision space. $n(t)$ varies in the optimization process. When $n(t)$ varies, the objective functions $\mathbf{F}$ must change in the same time. As no works haven been done for the time-dependent time-dependent constraints, no studies have been done on this type of dynamic characteristics.
- Time-dependent dimension of objective space. This is caused by adding or removing objective functions during the optimization process. Although changing number of objectives happens widely in real-life, this type of dynamic is nearly untouched.

The framework of dynamic handling technique for DMOPs is shown in Figure 2.2.The main steps of the dynamic handling technique are reconstruction, reproduction and selection.

When a change is detected, algorithms react to the changes by reconstructing the population. There are plenty of reconstructing strategies being proposed in the literature. The simplest idea is reconstructed the population with random solutions [9, 86, 87]. The works based on this method usually focus on two problems: 1) how many solution need to be reconstructed, 2) how to generate the new solutions. Some based on the prediction

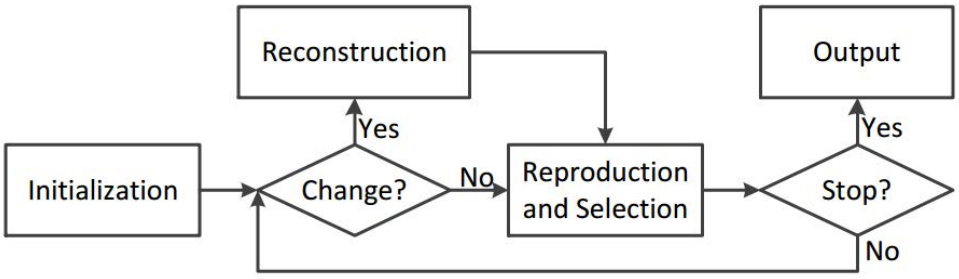


Figure 2.2: Framework of dynamic handling technique

[88–96]. According to the historical PF or PS, the algorithms use mathematical model to predict the new PF or PS. Then algorithms reconstructed the population around the predicted PF or PS. This type of methods performs well on handling the moving PS or PF. Some others based on the recorded previous optimal [97, 98], the algorithms set a "state" every time the problem is changed and recorded the best solution(s) for every state. Once the algorithms find the problem turns back to a recorded state, the recoded best solution(s) is/are used to reconstruct the population. This type of method performs well on the DMOPs with a periodical change, which means the PS/PF is expected to return to the same position in the future. Unlike the reproduction only change the population once, the reproduction and selection can continuously impact the population for a long term after the change. One typical example based on this method is the D-NSGA-II [9], which does not reconstruct the population after the changes, but replaces part of the population with random solutions in the selection.

2.4.2 Methods

In this section, we introduce the current dynamic handling method for DMOPs. Noted that only time-dependent objective functions have been studied. All these works are about this type of dynamics. The existing dynamic handling techniques can be classified into three categories:

- *Diversity enhancement*: The basic idea of this category is injecting new solutions into a converged population to increase the diversity. With the guidance of these

new solutions, the solutions in the population are likely to jump out of the current optimum position and move towards the new PF. Diversity enhancement can be divided into two subcategories according to the timing of injecting new solutions. One is responding to the dynamics after a change is detected, which is known as *diversity introduction*. Some works [9, 99, 100] propose hyper-mutation methods which help the solutions in the population jump out of their current positions, thus to handle the dynamics. In some other works [9,86,87], solutions generated either randomly or specifically by some heuristic methods are injected into the current population. All these new solutions participate in the survival competition together with the existing ones. The other subcategory is maintaining the population diversity during the whole process, which is called *diversity maintenance*. Unlike *diversity introduction*, *diversity maintenance* does not respond to dynamics explicitly. It mainly relies on the diversity maintenance strategy to track the moving PF. Multi-population strategy [101] is also designed to enhance the population diversity.

- *Memory mechanism*: The basic idea of memory mechanism is accelerating the convergence progress with the help of historical data. This method only works when the PS moves periodically, which means the PS is expected to return to the same position in the future. Wang et al. [97] have proposed a hybrid memory scheme which relocates the new population either by the previous non-dominated solution or by a local search on the archive. Peng et al. have [98] exploited the historical knowledge and uses it to build an evolutionary environment model. This model serves to guide the adaptation to the changing environment.

- *Prediction strategy*: The idea for this type is to predict the location of the new PS and rebuild the population in that location. It can be regarded as an upgraded version of memory mechanism. Instead of directly using the historical solutions, the algorithm uses historical information to predict the future location of the PS. For example, works from Hatzakis et al. [88, 89] employ an autoregressive model to

track the new PS population within the neighbourhood. By utilizing the regularity property [90], Zhou et al. [91, 92] and Peng et al. [93] have built a model for the movements of the PS manifolds' centres under different dynamic environments. Wu et al. [94] and Koo et al. [95] have proposed to predict the optimal moving direction of the population. More recently, Muruganantth et al. [96] have proposed a hybrid scheme, which uses either the Kalman Filter or random re-initialization based on their performance.

2.4.3 Performance Indicators

Most of the indicators for dynamic multi-objective optimization are extended from the indicators for multi-objective optimization. The indicators can be classified into three types. To be brief, we denoted the indicators for MOPs as static indicators.

- The performance at the end of optimization also can be used to indicate the performance for the algorithms handling DMOPs. Ignoring all the situations during the optimization process, the performance indicator only measure the final outcome. Any indicators developed for multi-objective optimization can be used. For example:

  Final IGD(FIGD): $P^*_{t_{max}}$ is a set of points uniformly sampled along the $PF_{t_{max}}$, $S_{t_{max}}$ is the set of solutions obtained by an EMO algorithm for approximating the $PF_{t_{max}}$, and t_{max} is the final time step in a run,

  $$\text{FIGD} = \text{IGD}(P^*_{t_{max}}, S_{t_{max}}). \tag{2.7}$$

- Mean value before changes calculates the mean value of the indicators developed for MOPs before every change. Indicators in this type mainly focus on how well the algorithms recover from the changes. For example:

  Mean IGD(MIGD): P^*_t is a set of points uniformly sampled along the PF_t, S_t is the set of solutions obtained by an EMO algorithm for approximating the PF_t, and T

is a set of time steps, which are one step before every changes in a run,

$$\text{MIGD} = \frac{1}{|T|} \sum_{t \in T} \text{IGD}(P_t^*, S_t). \tag{2.8}$$

- We also can calculate the integration of the value of an indicator during the whole optimization process, instead of using the value at time steps before changes. This type of performance indicators focuses on how quick the algorithms recover from the changes. For example:

  Integrated Inverted Generational Distances(IIGD): P_t^* is a set of points uniformly sampled along the PF_t, S_t is the set of solutions obtained by an EMO algorithm for approximating the PF_t.$t_{begin} \leq t \leq t_{end}$, t_{begin} and t_{end} is the starting time and ending time for the optimization process.

$$\text{IIGD} = \frac{1}{|T|} \int_{t_{begin}}^{t_{end}} \text{IGD}(P_t^*, S_t). \tag{2.9}$$

CHAPTER 3

DYNAMIC TWO-ARCHIVE EVOLUTIONARY ALGORITHM

Most existing research focus on the dynamic multi-objective optimization problems (DMOPs) with time-dependent objective functions, whereas DMOPs with a changing number of objectives have seldom been considered yet. DMOPs with a changing number of objectives can be found in many real-life scenarios, such as software project scheduling and operations management. Time-dependent objective functions usually lead to the change of shape or position of the Pareto-optimal front/set. However, the increase/decrease in the number of objectives always results in the expansion or contraction of the dimension of the Pareto-optimal front/set manifold. Unfortunately, most existing research can hardly handle this type of dynamics. This chapter begins with the definition of dynamic multi-objective optimization problems with a changing number of objectives. Then a dynamic two-archive evolutionary algorithm (DTAEA) is implemented to tackle this problem. DTAEA maintains two populations simultaneously. For each population, there are different updating and reconstructing strategies, with a mating strategy proposed. Last, seven other research questions will be discussed and analysed.

This chapter is organised in the following way. Section 3.1 clarifies the motivation. Section 3.2 probes DTAEA and its implementation. Experimental results of DTAEA are presented in Section 3.3. Further research questions and analysis of the DTAEA are carried out in Section 3.4. Finally, Section 3.6 summarises this chapter.

3.1 Motivation

Multi-objective optimization, which can often be found in real-life scenarios (e.g. engineering [102], economics [103], and logistics [104]), tries to achieve the trade-off among two objectives or more simultaneously. Due to the conflicting nature of the objectives, it is impossible to obtain a solution optimizing all objectives. Instead of finding one solution, the best trade-off solutions among those objectives, known as Pareto optimal solutions, have aroused the interest of the decision makers. In real-life optimization scenarios, a dynamic incident happens uncertainly. For example, in a production line, new jobs arrive over time and machines in the production line breakdown from time to time [105]. Another example can be found in a design process. The quality of the raw material changes in the production process [106]. Generally speaking, dynamic optimization problems are usually more challenging than those stationary ones, because the algorithm needs not only to achieve trade-off solutions, but also to deal with the changes. An effective dynamic multi-objective optimization algorithm should be able to track the time-dependent Pareto-optimal front. Evolutionary Algorithm (EA) is known as a powerful and popular tool to handle dynamic multi-objective optimization [8, 9, 12, 99, 100]. The existing dynamic handling techniques can be classified into three categories:

- *Diversity enhancement*: Diversity includes two aspects: 1) cover the whole Pareto-optimal front (PF), 2) well spread the population on the PF. The basic idea of this category is injecting new solutions into a converged population to increase the diversity. With the guidance of these new solutions, the solutions in the population are likely to move twards the new Pareto-optimal front from the current optimum position. Diversity enhancement can be divided into two subcategories according to the timing of injecting new solutions. One is responding to the dynamics after a change is detected, which is known as *diversity introduction*. It is usually accompanied with a change detection technique. Previous works [9, 99, 100] propose some hypermutation methods which help the solutions in the population jump out of their

current positions, thus to handle the dynamics. In some works [9, 86, 87], solutions generated either randomly or specifically by some heuristic methods are injected into the current population. All these new solutions participate in the survival competition together with the existing ones. The other subcategory is maintaining the population diversity during the whole process, which is called *diversity maintenance*. Unlike *diversity introduction*, *diversity maintenance* does not respond to dynamics explicitly. It mainly relies on the diversity maintenance strategy to track the moving Pareto-optimal front. Multi-population strategy [101] has been also designed to enhance the population diversity.

- *Memory mechanism*: The basic idea of memory mechanism is accelerating the convergence progress with the help of historical data. The algorithms based on memory mechanism record the state of the Pareto-optimal set (PS) and the best solution(s) for this PS before changes. Once the problem return to the same PS, algorithms reconstruct the population with the recorded solution(s). This method only works when the Pareto-optimal set moves periodically, which means the PS is expected to return to the same position in the future. Wang et al. [97] proposes a hybrid memory scheme which relocates the new population either by the previous non-dominated solution or by a local search on the archive. Peng et al. [98] exploits the historical knowledge and uses it to build an evolutionary environment model. This model serves to guide the adaptation to the changing environment.

- *Prediction strategy*: The idea for this type is to predict the location of the new Pareto-optimal set and rebuild the population in that location. It can be regarded as an upgraded version of memory mechanism. Instead of directly using the historical solutions, the algorithm uses historical information to predict the future location of the Pareto-optimal set. For example, Hatzakis et al. [88,89] employ an autoregressive model to track the new Pareto-optimal set population within the neighbourhood. By utilizing the regularity property [90], Zhou et al. [91, 92] and Peng et al. [93]

have built model for the movements of the Pareto-optimal set manifolds' centres under different dynamic environments. Wu et al. [94] and Koo et al. [95] propose algorithms predict the optimal moving direction of the population by applying a local search around the predicted Pareto-optimal set maniforlds' centere. More recently, Muruganantha et al. [96] proposes a hybrid scheme, which uses either the Kalman Filter or random re-initialization based on their performance.

It is worth noting that the current studies on dynamic multi-objective optimization mainly focus on problems with time-dependent objective functions, which results in the change of the Pareto-optimal front or Pareto-optimal set over time. In addition to this type of dynamics, the number of objective functions may also change over time in many real-life scenarios. Changing the number of objective functions can be found in a wide range of situations, such as:

- The number of objective functions changes naturally as a requirement from the real-life optimization scenarios. For example, in the model of Rational Unified Process (RUP) software project scheduling problem, the number of quality function changes as the project goes onto different phases, resulting in changing number of objectives. Similar situations also happen in the cloud computing resource scheduling, where more varieties of objectives are considered. The quality of service (QoS) of each user increases or decreases when users come and leave.

- The number of objective functions changes as expected by the decision maker (DM). The example can be found in business operations management. The performance objectives of operations management include quality, speed, flexibility, dependability and cost [10]. The optimization process tries to achieve a trade-off among these objectives, and the DM is allowed to pick up or discard any objective during the optimization process based on stage results.

- Modelling real-life optimization scenarios is normally very difficult. It is unrealistic to request a thoroughly considered optimization problem before the optimization

process starts. Allowing the model builder to change the number of objectives during the optimization can reduce the difficulty of modelling. The model builder can bear less pressure in the first modelling as they still have chances to change afterwards.

Unfortunately, there is a research gap in how to handle the problems with a changing number of objectives, which is only mentioned in a few works [15, 107, 108]. Guan et al. [107] is the only one who has discussed the effects of objective increment and replacement. The proposed method in it is simply to evaluate the solutions again after changes, which does nothing on the dynamics.

The dynamic multi-objective optimization problems (DMOPs) with a changing number of objectives have two major characteristics: 1) the Pareto-optimal front/set before the change is a subset of the one after increasing the number of objectives, and vice versa; and 2) instead of changing the position or the shape of the original Pareto-optimal front/set, increasing or decreasing the number of objectives usually results in the expansion and contraction of the dimension of the Pareto-optimal front/set manifold.

In order to understand the challenges posed by the changing number of objectives, the DTLZ2 [64] as the benchmark problem and multi-objective EA based on decomposition (MOEA/D) [43] as the baseline algorithm is used as a example. Note that the DTLZ series problems are scalable to any number of objectives and MOEA/D have been reported to show strong performance on a wide range of problems [43, 109, 110].

When Increasing the Number of Objectives

First, a good approximation to the Pareto-optimal front of two-objective DTLZ2 is obtained from a 200 generations' run with the MOEA/D [111]. The result is shown in Figure 3.1(a). Then, the underlying problem changes to a three-objective instance. Figure 3.2(b) shows the distribution of the population gained after re-evaluating the objective values. It can be seen that although the population still converges well to the Pareto-optimal front, the diversity is not satisfying any longer after the number of objectives increases.

The population locates on a curve. As shown in Figure 3.1(b), the population crowds on a curve.

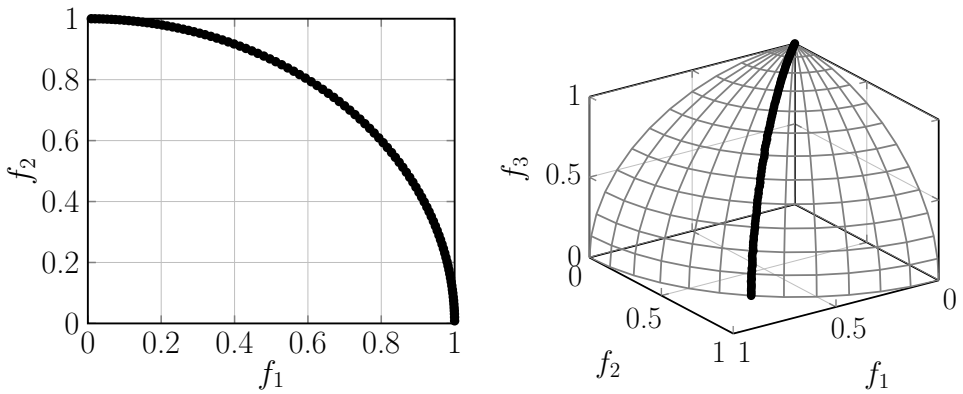
In general, the Pareto-optimal front/set manifold will expand after the number of the objectives increases. Also, the convergence of the population will not be affected, while the diversity will be significantly damaged. Therefore, the challenge for increasing number of objectives is how to push the population out of its original position.

A simple explanation for this feature is as followed. Let n to be the dimension of the decision space and the decision variables can be classified into two types: 1)diversity variables which only impacted location of a solution on the Pareto-optimal front, 2)convergence variables which only impacted the distance from a solution to the Pareto-optimal front(The general decision variables situation is discussed in Section 3.4.6). A optimization problem with m conflicted objectives have a $m-1$ dimension manifold PF. As a result, $m-1$ variables are the diversity variables, while $n-m+1$ variables are the convergence variables. When the algorithm achieve a good approximation, the $n-m+1$ convergence variables converged to a value, while the rest $m-1$ variables are well distributed. When m increased, the number of convergence variables decrease, which means some convergence variables turn to be diversity variables. The convergence of the population is not impacted because all the convergence variables are not impacted. As some part of convergence variables turn to diversity variables and they are converged to a value, the diversity of the population is bad.

When Decreasing the Number of Objectives

The existing techniques are not designed for the changing number of objectives. The pitfalls are listed as followed:

- Diversity enhancement techniques inject/maintain some diversity solutions in order to handle the dynamics. However, in a circumstance where number of objectives increases, these new solutions is unlikely to affect the previous population. The reason is that the convergence of the existing solutions in the population is not



(a) Approximation in the 2-objective case (b) Distribution of the population after increasing an objective

Figure 3.1: Comparison of population distributions when increasing the number of objectives.

affected, and the injected candidates is unlikely to survive inthe competition. On the other hand, when the number of objectives decreases, there is no extra selection pressure towards the Pareto-optimal front. Also, injecting solutions again cannot push the duplicated solutions out of the same reason in the situation of increasing number of objectives.

- Memory mechanism have difficulty in handling the problem because this problem is always not periodical. When the number of objective is periodical, the problem still may not be periodical. For example, we have objectives A and B at the beginning, then add objective C, and at last remove objective B. Although the number of objective is the same (two objectives), but it is not considered as the same state for the memory mechanism. For a problem with the number of objectives vary from m_1 to m_2, the possibility of returning to the same state is $\frac{1}{C_{m_2}^{m_1}}$. Here it should be pointed out that the memory mechanism has been used in [107] to handle the DMOPs with a changing number of objectives, which simply does nothing towards the changes but re-evaluates the population when the change of the environment is detected. Obviously, this strategy is too simple to handle problems with complicated properties.

- The prediction strategy is an effective method for DMOPs. Unfortunately, the

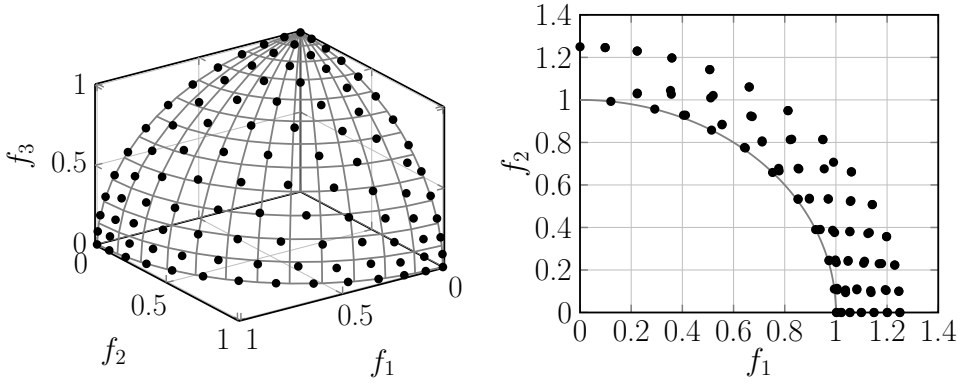
existing prediction strategies are usually proposed to track the movement of the Pareto-optimal set. The change of the number of objectives leads to the expansion or contraction of the Pareto-optimal set manifold, rather than a moving Pareto-optimal set. As a result, current algorithms can hardly handle this problem.

MOEA/D is run on a three-objective DTLZ2 instance for 300 generations. All parameters are set the same as the situation of increasing number of objectives. A reasonably good approximation to the Pareto-optimal front is obtained which is shown in figure3.2(a). Then, the underlying problem changes to a two-objective instance. Figure 3.2(b) shows the corresponding population distribution after re-evaluating the objective values. In contrast to the situation of increasing number of objectives, only some solutions stay on the Pareto-optimal front, while most of the solutions are located in the search spaces (as shown in Figure 3.2(b)).

Generally speaking, the effect of decreasing the number of objectives is the contraction of the Pareto-optimal front/set manifold. In addition, some solutions still stay on the Pareto-optimal front after the change, while the majority is moved away from it. Moreover, the diversity of the solutions on the Pareto-optimal front is not satisfactory. The population includes many duplicates or similar solutions. (as shown in figure3.2(b)). The challenges for decreasing number of objectives are 1) pulling the drifting solutions back to the Pareto-optimal front and 2) pushing the duplicated solutions out from their position.

3.2 Implementation

Bearing characteristics of DMOPs with a changing number of objectives in mind, this section presents a dynamic two-archive EA (denoted as DTAEA) to handle the DMOPs with a changing number of objectives. It should be pointed out that the multi-population strategy is not a brand new technique in the evolutionary multi-objective optimization literature. Works from Pradit et al. [6] and Wang et al. [7] are the two-archive evolu-



(a) Approximation in the 3-objective case. (b) Population distribution after decreasing an objective.

Figure 3.2: Comparison of population distributions when decreasing the number of objectives.

tionary algorithms (TAEA), in which two complementary archives are used to balance the convergence and diversity of the search process. But none of the existing two-archive algorithms can handle dynamic situations.

The high-level flow chart is shown in Figure 3.3. Similar to [7, 112, 113], DTAEA maintains two co-evolving populations simultaneously: the convergence archive (CA) constantly prefer convergent solutions which provide competitive selection pressure towards the PF. The diversity archive (DA) is used to provides extra information which maintains diversified solutions as much as possible. The size of the CA and the DA is equal and fixed to a constant N. In each iteration, if there is no change occurs, parents will be selected from CA and DA according to Section 3.2.4. Then CA and DA select solutions from the offspring based on their own selecting strategies (described in Section 3.2.3). If the number of objective changes, the CA and the DA are reconstructed according to the mechanism described in Section 3.2.2. CA are used as the final output population.

3.2.1 Basic Definitions

There are various types of dynamic characteristics that can result in different mathematical definitions. The focus of this research mainly stay on the continuous DMOPs defined

as follows:

$$\begin{aligned} &\text{minimize} \quad \mathbf{F}(\mathbf{x}, t) = (f_1(\mathbf{x}, t), \cdots, f_{m(t)}(\mathbf{x}, t))^T \\ &\text{subject to} \quad \mathbf{x} \in \Omega, t \in \Omega_t \end{aligned} \tag{3.1}$$

where t is a discrete time point defined as $t = \lfloor \frac{\tau}{\tau_t} \rfloor$, τ and $\frac{1}{\tau_t}$ represent the iteration counter and the frequency of change, respectively, and $\Omega_t \subseteq \mathbb{N}$ is a set of the time points. $\Omega = \prod_{i=1}^{n}[a_i, b_i] \subseteq \mathbb{R}^n$ is the decision (variable) space, $\mathbf{x} = (x_1, \cdots, x_n)^T \in \Omega$ is a candidate solution. $\mathbf{F} : \Omega \to \mathbb{R}^{m(t)}$ constitutes of $m(t)$ real-valued objective functions and $\mathbb{R}^{m(t)}$ is called the objective space at time step t, where $m(t)$ is a discrete function of t.

Definition 1. *At time step t, $\mathbf{x}^1$ is said to Pareto dominate $\mathbf{x}^2$, denoted as $\mathbf{x}^1 \preceq_t \mathbf{x}^2$, if and only if: $\forall i \in \{1, \cdots, m(t)\}$, $f_i(\mathbf{x}^1, t) \leq f_i(\mathbf{x}^2, t)$; and $\exists j \in \{1, \cdots, m(t)\}$, $f_j(\mathbf{x}^1, t) < f_j(\mathbf{x}^2, t)$.*

Definition 2. *At time step t, $\mathbf{x}^* \in \Omega$ is said to be Pareto-optimal, if there is no other $\mathbf{x} \in \Omega$ such that $\mathbf{x} \preceq_t \mathbf{x}^*$.*

Definition 3. *At time step t, the set of all Pareto-optimal solutions is called the t-th Pareto-optimal set (PS_t). The corresponding set of Pareto-optimal objective vectors is called the t-th Pareto-optimal front (PF_t), i.e., $PF_t = \{\mathbf{F}(\mathbf{x}, t) | \mathbf{x} \in PS_t\}$.*

Definition 4. *At time step t, the t-th ideal objective vector is $\mathbf{z}^*(t) = (z_1^*(t), \cdots, z_{m(t)}^*(t))^T$, where $z_i^*(t) = \min_{\mathbf{x} \in \Omega} f_i(\mathbf{x}, t), i \in \{1, \cdots, m(t)\}$.*

Definition 5. *At time step t, the t-th nadir objective vector is $\mathbf{z}^{nad}(t) = (z_1^{nad}(t), \cdots, z_{m(t)}^{nad}(t))^T$, where $z_i^{nad}(t) = \max_{\mathbf{x} \in \Omega} f_i(\mathbf{x}, t), i \in \{1, \cdots, m(t)\}$.*

Definition 6. *Under some mild conditions, the regularity property, induced from the Karush-Kuhn-Tucker condition, means that the PF and PS of a continuous m-objective MOP is an $(m-1)$-dimensional piecewise continuous manifold in both the objective space and the decision space.*

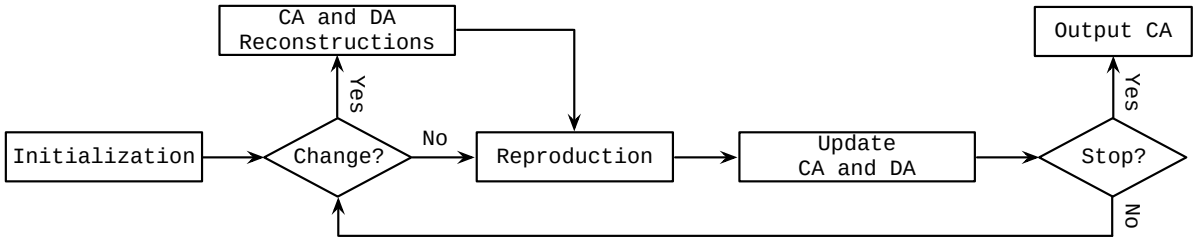


Figure 3.3: Flow chart of DTAEA.

3.2.2 Reconstruction Mechanisms

The reconstruction is the crucial part of the algorithm. The basic idea is to relocate or replace solutions in CA and DA. Since the location of solutions after increasing and decreasing the number of objectives is different (discussed in Section 3.1), the reconstruction mechanisms are different accordingly.

When Increasing the Number of Objectives

As discussed in Section 3.1, the convergence of the population is not affected when increasing the number of objectives, while the population diversity is unsatisfied. As all original solutions in CA before increasing the number of objectives are possibly on or close to the new PF, CA before the change will be constructed as the new CA. This avoids the loss of convergence before finding any better solutions. Meanwhile, all solutions in DA are discarded, as they are no longer considered as good in terms of diversity. The randomly generated solutions will be filled in new DA. In particular, the canonical Latin hypercube sampling (LHS) method [114] is employed to uniformly sample N random solutions from the decision space. The pseudo-code of this reconstruction mechanism is given in Algorithm 1.

When Decreasing the Number of Objectives

As discussed in Section 3.1, the solutions from the original set can be classified into two types. One is the convergent and duplicated solutions. The other is solutions drifted away from the PF. The solutions from the first type in CA can be considered as still close to new

Algorithm 1: Reconstruction Mechanism After Increasing the Number of Objectives

Input: The last CA before decreasing the number of objectives P
Output: CA, DA
CA $\leftarrow$ P;
Use the LHS to sample N random solutions to form the DA;
return CA and DA

PF, therefore these solutions will be kept in the new CA. If the new CA is not full, CA will be filled up with polynomial mutated [115] solutions which generated from some selected competitive non-dominated solutions in CA. In particular, the non-dominated solutions used for mutation are chosen by a binary tournament selection, where the non-dominated solution located in a area with lower density is preferred. Since solutions drifted away from the PF are spread in the search space, all the dominated solutions in the last CA before decreasing the number of objectives will be kept in the new DA. To help the solutions jump out from their duplicates, the canonical LHS method is used to generate $N - |DA|$ random solutions to fill the gap in the new DA,$|DA|$ is the number of solutions in the DA. The pseudo-code of this reconstruction mechanism is given in Algorithm 2. The Binary Tournament selection is described in Algorithm 3.

Algorithm 2: Reconstruction Mechanism After Decreasing the Number of Objectives

Input: The last CA before increasing the number of objectives P
Output: CA, DA
CA $\leftarrow$ `NonDominationSelection`(P); `// choose all non-dominated solutions`
DA $\leftarrow$ P $\setminus$ CA;
while |CA| $< N$ **do**
    $\mathbf{x}^c \leftarrow$ `BinaryTournamentSelection`(CA);
    $\mathbf{x}^m \leftarrow$ `PolynomialMutation`(x^c);
    CA $\leftarrow$ CA $\bigcup \{\mathbf{x}^m\}$;
Use the LHS to sample $N -$ |DA| random solutions to fill the DA;
return CA and DA

To facilitate the density estimation, N uniformly distributed weight vectors, i.e., $\mathrm{W}(t) = \{\mathbf{w}^1(t), \cdots, \mathbf{w}^N(t)\}$ in $\mathbb{R}^{m(t)}$ should be given. In particular, we employ the weight vector generation method developed by Li et al. [75] for this purpose, since it is scalable

Algorithm 3: Binary Tournament Selection

Input: Solution set S
Output: Selected solution $\mathbf{x}^c$

Randomly select two solutions $\mathbf{x}^1$ and $\mathbf{x}^2$ from S;
if $\texttt{Density}(\mathbf{x}^1) < \texttt{Density}(\mathbf{x}^2)$ **then**
  $\mathbf{x}^c \leftarrow \mathbf{x}^1$;
else if $\texttt{Density}(\mathbf{x}^1) > \texttt{Density}(\mathbf{x}^2)$ **then**
  $\mathbf{x}^c \leftarrow \mathbf{x}^2$;
else
  $\mathbf{x}^c \leftarrow$ Randomly pick one between $\mathbf{x}^1$ and $\mathbf{x}^2$;
return $\mathbf{x}^c$

to the many-objective scenarios. Accordingly, these weight vectors divide $\mathbb{R}^{m(t)}$ into N subspaces, i.e., $\Delta^1(t), \cdots, \Delta^N(t)$. In particular, a subspace $\Delta^i(t)$, where $i \in \{1, \cdots, N\}$ is defined as:

$$\Delta^i(t) = \{\mathbf{F}(\mathbf{x}, t) \in \mathbb{R}^{m(t)} | \langle \mathbf{F}(\mathbf{x}, t), \mathbf{w}^i(t) \rangle \leq \langle \mathbf{F}(\mathbf{x}, t), \mathbf{w}^j(t) \rangle\} \tag{3.2}$$

where $j \in \{1, \cdots, N\}$ and $\langle \mathbf{F}(\mathbf{x}, t), \mathbf{w}(t) \rangle$ is the perpendicular distance between $\mathbf{F}(\mathbf{x}, t)$ and the reference line formed by the origin and $\mathbf{w}(t)$. After the setup of subspaces, each solution of the underlying population is associated with an unique subspace according to its position in $\mathbb{R}^{m(t)}$. Specifically, for a solution $\mathbf{x}$, the index of its associated subspace is determined as:

$$I(\mathbf{x}) = \underset{i \in \{1, \cdots, N\}}{\operatorname{argmin}} \langle \overline{\mathbf{F}}(\mathbf{x}, t), \mathbf{w}^i(t) \rangle \tag{3.3}$$

where $\overline{\mathbf{F}}(\mathbf{x}, t)$ is the normalized objective vector of $\mathbf{x}$, and its i-th objective function is calculated as:

$$\overline{f}_i(\mathbf{x}, t) = \frac{f_i(\mathbf{x}, t) - z_i^*(t)}{z_i^{nad}(t) - z_i^*(t)} \tag{3.4}$$

where $i \in \{1, \cdots, m(t)\}$. Based on the association relationship between solutions and subspaces, the density of a subspace is evaluated as the number of its associated solutions. The Density estimation for a solution $\mathbf{x}$ is defined as:

$$\texttt{density}(\mathbf{x}) = |\{\mathbf{x}^* \in \mathbf{S} | Id(\mathbf{x}) = I(\mathbf{x}^*)\}| \tag{3.5}$$

in which S is the solution set.

Figure 3.4 gives a simple example to illustrate this density estimation method. In particular, five weight vectors divide the underlying objective space into five subspaces, where the corresponding density of each subspace is 2, 3, 1, 1 and 3, respectively.

3.2.3 Update Mechanisms

The update mechanisms are also different and will be described separately.

Update Mechanism of the CA

CA tries to provides continuously selection pressure towards the PF. As a result, CA prefers those non-dominated solutions. At the beginning, the fast non-dominated sorting method developed by Deb et al. [32] to divide the mixed population (combination of CA and offspring) R into several non-domination levels, i.e., F_1, F_2 and so on. Starting from F_1, each non-domination level is filled into the new CA. This procedure continues until the size of the new CA equals to or exceeds the population size. Here the last included non-domination level is denoted as F_l, while all solutions from F_{l+1} onwards are discarded. If the size of the new CA equals to population size, the update procedure terminates. Otherwise, the density estimation method introduced in Section 3.2.2 is used to evaluate the diversity information of the current CA. Then, the worst solution in the most crowded subspace will be removed from the CA until the size of CA is equal to the population size. In particular, for a subspace $\Delta^i(t)$, the worst solution $\mathbf{x}^w$ is defined as:

$$\mathbf{x}^w = \underset{\mathbf{x}\in\Delta^i(t)}{\operatorname{argmax}}\{g^{tch}(\mathbf{x}|\mathbf{w}^i(t), \mathbf{z}^*(t))\} \tag{3.6}$$

where the Tchebychev distance g^{tch} is calculated as:

$$g^{tch}(\mathbf{x}|\mathbf{w}^i(t), \mathbf{z}^*(t)) = \max_{1\leq j\leq m(t)}\{|f_j(\mathbf{x}, t) - z_j^*(t)|/w_j^i(t)\}. \tag{3.7}$$

The pseudo-code of the update mechanism of the CA is given in Algorithm4. Figure 3.4 gives an example to illustrate this update mechanism. Assume that the black and green circles indicate the solutions of the original CA and the offspring population respectively. Since the first non-domination level already contains 8 solutions, $\mathbf{x}^3$ and $\mathbf{x}^7$, which belong to the second non-domination level, are not considered any longer. In the first iteration, the worst solution is $\mathbf{x}^8$ because $\Delta^5(t)$ is most crowded subspace and $g^{tch}(\mathbf{x}^9) > g^{tch}(\mathbf{x}^8)$. $\mathbf{x}^8$ is removed from the CA. Analogously, $\mathbf{x}^2$ and $\mathbf{x}^9$ will be eliminated in the latter iterations.

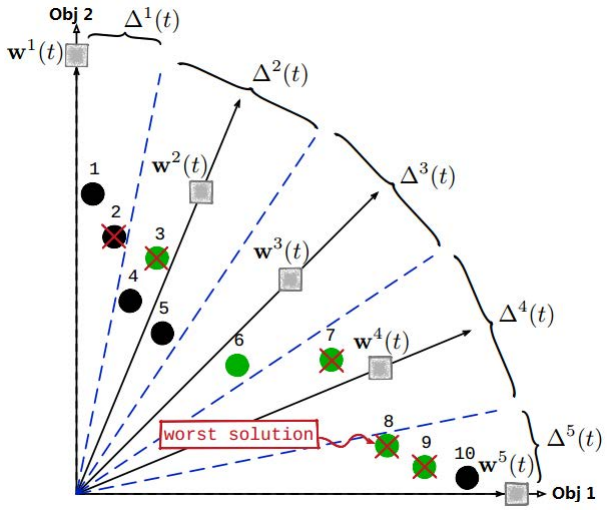


Figure 3.4: An example of the CA's update mechanism. Note that $\mathbf{x}^i$ is denoted as its index i for short.

Update Mechanism of the DA

The DA is providing extra information which improves the diversity. That means the DA should maintain diversified solutions in the areas under exploited by the CA. Similar to the update mechanism of the CA, the combination of the DA and the offspring population is denoted as R; in the meanwhile, CA will be the reference set. Afterwards, the density estimation method introduced in Section 3.2.2 builds up the association relationship between solutions in R and the subspaces. Then, based on the associated relationship, for each subspace, the mechanism try to keep I_{tr} $(1 \leq \mathsf{I}_{\mathsf{tr}} \leq N)$ solutions at the I_{tr}-th

Algorithm 4: Update Mechanism of the CA

Input: CA, offspring population Q, weight vector set W
Output: Updated CA

$\mathrm{S} \leftarrow \emptyset$, $i \leftarrow 1$;
$\mathrm{R} \leftarrow \mathrm{CA} \bigcup \mathrm{Q}$;
$\{F_1, F_2, \cdots\} \leftarrow \texttt{NonDominatedSorting}(\mathrm{R})$;
while $|\mathrm{S}| \leq N$ **do**
    $\mathrm{S} \leftarrow \mathrm{S} \bigcup F_i$, $i \leftarrow i + 1$;
if $|\mathrm{S}| = N$ **then**
    $\mathrm{CA} \leftarrow \mathrm{S}$;
else
    foreach $\mathbf{x} \in \mathrm{S}$ **do**
        $\overline{\mathbf{F}}_k(\mathbf{x}, t) = \frac{\mathbf{F}(\mathbf{x},t) - \mathbf{z}^*(t)}{\mathbf{z}^{nad}(t) - \mathbf{z}^*(t)}$;
    $\{\Delta^1(t), \cdots, \Delta^{|\mathrm{W}|}(t)\} \leftarrow \texttt{Association}(\mathrm{S}, \mathrm{W})$;
    while $|\mathrm{S}| > N$ **do**
        Find the most crowded subspace $\Delta^i(t)$;
        $\mathbf{x}^w \leftarrow \underset{\mathbf{x} \in \Delta^i(t)}{\operatorname{argmax}}[g^{tch}(\mathbf{x}|\mathbf{w}^i(t), \mathbf{z}^*(t))]$;
        $\mathrm{S} \leftarrow \mathrm{S} \setminus \{\mathbf{x}^w\}$;
    $\mathrm{CA} \leftarrow \mathrm{S}$;
return CA

iteration. In particular, at the I_{tr}-th iteration, if the CA already has I_{tr} solutions or there is no solution in R associated with the currently investigating subspace, this subspace will be ignored and then move to the next subspace. Otherwise, the best non-dominated solution in R associated with this currently investigating subspace is chosen to be added into the newly formed DA. In particular, the best solution $\mathbf{x}^b$ of the currently investigating subspace $\Delta^c(t)$ is defined as:

$$\mathbf{x}^b = \underset{\mathbf{x} \in \mathrm{O}}{\operatorname{argmin}}\{g^{tch}(\mathbf{x}|\mathbf{w}^c(t), \mathbf{z}^*(t))\} \tag{3.8}$$

where O constitutes of the non-dominated solutions in R associated with $\Delta^c(t)$. This iterative investigation continues until the DA is filled to the given size. The pseudo-code of the update mechanism of the DA is given in Algorithm 6. Figure reffig:DTAEA:updateDA gives an example to illustrate this update mechanism. Assume that the grey triangles represent the solutions of the updated CA while the black and green circles indicate the

Algorithm 5: Association Operation

Input: Solution set S, weight vector set W
Output: Subspaces $\Delta^1(t), \cdots, \Delta^{|\mathrm{W}|}(t)$

```
foreach x ∈ S do
    foreach w ∈ W do
        Compute d^⊥(x, w) = x − w^T x/‖w‖;
    k ← argmin_{w∈W} d^⊥(x, w);
    Δ^k(t) ← Δ^k(t) ⋃ {x};
return Δ^1(t), ··· , Δ^|W|(t)
```

solutions of the original DA and the offspring population. At the first iteration, the update mechanism of the DA starts from $\Delta^1(t)$. Since the CA already has two solutions in this subspace, we move to investigate $\Delta^2(t)$. As the CA does not have any solution in this subspace, the non-dominated solution with highest g^{tch} in this subspace, i.e., $\mathbf{x}^3$ is included into the newly formed DA. Thereafter, since $\Delta^3(t)$ to $\Delta^5(t)$ already have a solution in the CA, they are not considered during this iteration. At the second iteration, since the CA already has two solutions in $\Delta^1(t)$, it is still not considered during this iteration. As for $\Delta^2(t)$, since the CA does not have any solution in this subspace, the remaining best solution $\mathbf{x}^5$ is selected this time. Then, since $\Delta^3(t)$ to $\Delta^5(t)$ only have one solution in the CA, we can choose the best solution from R to be included into the newly formed DA this time. At the end of the second iteration, the DA is filled and the update procedure terminates.

3.2.4 Offspring Reproduction

The interaction between the CA and the DA makes DTAEA work. The mating selection mechanism should choose the mating parents from the CA and the DA respectively according to solution distribution of the CA and the DA. CA provides continuously selection pressure towards the PF, as a result, the mechanism should always select solutions from CA. The DA is an auxiliary for the CA which maintaining diversified solutions, solutions should be selected when the diversity of CA is unsatisfied. It should be pointed out that

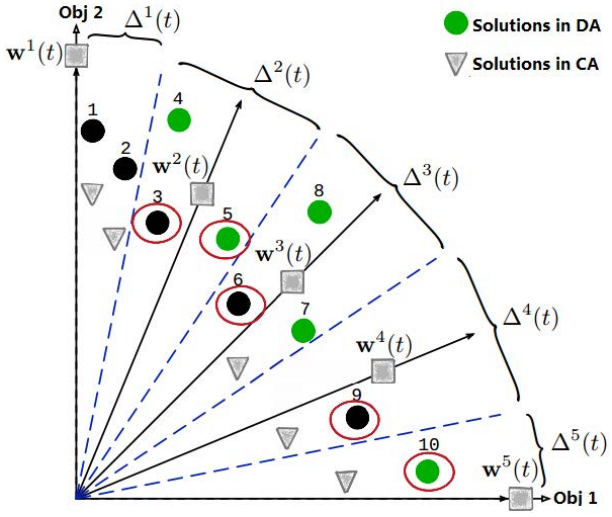


Figure 3.5: An example of the DA's update mechanism. Note that $\mathbf{x}^i$ is denoted as its index i for short.

the CA and the DA become gradually the same, but with different convergent speed. This results in a weakened complementary effects between these two co-evolving populations. According to those factors, the pseudo code of the restricted mating selection mechanism is given in Algorithm 7. The first mating parent is always randomly selected from the CA. If the diversity of the CA is not good (according to the the occupation rate), the second mating parent is randomly chosen from the DA; otherwise, it still comes from the CA. The diversity of the CA is measured by the occupation rate (denoted as I^o_{CA}) which is the percentage of the subspaces which have solutions.

$$I^o_{\text{CA}}(t) = 1 - \frac{|\{\Delta^i((t)|1 \leq i \leq |W|, \Delta^i = \emptyset\}|}{|W|} \tag{3.9}$$

In which $|\cdot|$ operation calculates the size of a set. It is intuitive to understand that a high I^o_{CA} indicates a well-diversified distribution in the CA. Accordingly, we should give a larger chance to select the second mating parent from the CA. Based on the selected mating parents, the offspring solution can be generated by the appropriate crossover and mutation operations as desired.

Algorithm 6: Update Mechanism of the DA

Input: CA, DA, offspring population Q, weight vector set W
Output: Updated DA

$\mathrm{S} \leftarrow \emptyset$, $i \leftarrow 1$;
$\mathrm{R} \leftarrow \mathrm{DA} \bigcup \mathrm{Q}$;
$\{\Delta^1(t), \cdots, \Delta^{|\mathrm{W}|}(t)\} \leftarrow \texttt{Association}(\mathrm{R}, \mathrm{W})$;
$\mathsf{itr} \leftarrow 1$;
while $|\mathrm{S}| \leq N$ **do**
  for $i \leftarrow 1$ **to** $|\mathrm{W}|$ **do**
    if $\Delta^i(t) \neq \emptyset \wedge$ CA *has less than* itr *solutions in* $\Delta^i(t)$ **then**
      $\mathrm{O} \leftarrow \texttt{NonDominationSelection}(\Delta^i(t))$;
      $\mathbf{x}^b \leftarrow \underset{\mathbf{x} \in \mathrm{O}}{\operatorname{argmin}} \{g^{tch}(\mathbf{x} | \mathbf{w}^c(t), \mathbf{z}^*(t))\}$;
      $\Delta^i(t) \leftarrow \Delta^i(t) \setminus \{\mathbf{x}^b\}$;
      $\mathrm{S} \leftarrow \mathrm{S} \bigcup \{\mathbf{x}^b\}$;
  $\mathsf{itr} \leftarrow \mathsf{itr} + 1$;
$\mathrm{DA} \leftarrow \mathrm{S}$;
return DA

Algorithm 7: Mating Selection Mechanism

Input: CA, DA
Output: Mating parents $\mathbf{p}_1$, $\mathbf{p}_2$

$\mathbf{p}_1 \leftarrow \texttt{RandomSelection}(\mathrm{CA})$;
if $\mathsf{rnd} < I^o_{\mathrm{CA}}$ **then**
  $\mathbf{p}_2 \leftarrow \texttt{RandomSelection}(\mathrm{CA})$;
else
  $\mathbf{p}_2 \leftarrow \texttt{RandomSelection}(\mathrm{DA})$;
return $\mathbf{p}_1$, $\mathbf{p}_2$

3.2.5 Time Complexity Analysis

This subsection discusses the complexity of DTAEA in one generation. Since the CA and the DA share the same initial population, the initialization procedure costs $\mathcal{O}(N)$ function evaluations, where N is the population size. The CA and the DA will be reconstructed once the the number of objectives changes. For the situation of the number of objectives increases or decrease, all solutions should be re-evaluated, the reconstruction of the CA and and the DA both cost $\mathcal{O}(N)$ function evaluations. When decreasing the number of objectives, the non-dominated solutions costs $\mathcal{O}(N^2)$ comparisons. In the update mechanism of the CA, the non-dominated sorting (line 3 of Algorithm 4) and the association

operation (line 11 of Algorithm 4) cost $\mathcal{O}(N^2)$ comparisons and $\mathcal{O}(N|W|)$ respectively, where $|W|$ returns the cardinality of a W. As for the update mechanism of the DA, the association operation in line 3 of Algorithm 6 costs $\mathcal{O}(N|W|)$ comparisons. During the main while loop, the update procedure performs one by one for the subspaces. In particular, if $\Delta^i(t) \neq \emptyset$, where $i \in \{1, \cdots, |W|\}$, the most time consuming part of the for loop from line 6 to line 11 is the non-dominated sorting in line 8 of Algorithm 6, which costs $\mathcal{O}(|\Delta^i(t)| \log |\Delta^i(t)|)$ comparisons. Since $\sum_{i=1}^{|W|} \Delta^i(t) = 2N$, this for loop costs $\mathcal{O}(N^2)$ comparisons in total. In summary, the complexity of DTAEA in one generation is $\mathcal{O}(N^2)$.

3.3 Results

This section contains the experimental result.

3.3.1 Benchmark Problems

Benchmark problems play important roles in assessing and analysing the performance of an algorithm, and thus guiding its further developments. Although some dynamic multi-objective benchmark problems have been proposed in the literature [116–118], most, if not all, of them merely consider the dynamically changing shape or position of the PF or PS. In the work from Huang et al, [15], a dynamic multi-objective test instance with a changing number of objectives was developed, for the first time, by modifying a particular test instance from the classic DTLZ benchmark suite [64]. In this paper, we develop a series of dynamic multi-objective benchmark problems as shown in Table 3.1. Note that, in addition to the changing number of objectives, F5 and F6 are also accompanied by a time-dependent change of the shape or position of the PF or PS.

Table 3.1: Mathematical Definitions of Dynamic Multi-Objective Benchmark Problems

Problem Instance	Definition	Domain
F1	$f_1 = (1+g)0.5\prod_{i=1}^{m(t)-1} x_i$ $f_{j=2:m(t)-1} = (1+g)0.5(\prod_{i=1}^{m(t)-j} x_i)(1-x_{m(t)-j+1})$ $f_{m(t)} = (1+g)0.5(1-x_1)$ $g = 100[n-m(t)+1+\sum_{i=m(t)}^{n}((x_i-0.5)^2-\cos(20\pi(x_i-0.5)))]$	$[0,1]$
F2	$f_1 = (1+g)0.5\prod_{i=1}^{m(t)-1}\cos(x_i\pi/2)$ $f_{j=2:m(t)-1} = (1+g)0.5(\prod_{i=1}^{m(t)-j}\cos(x_i\pi/2))(\sin(x_{m(t)-j+1}\pi/2))$ $f_{m(t)} = (1+g)\sin(x_1\pi/2)$ $g = \sum_{i=m(t)}^{n}(x_i-0.5)^2$	$[0,1]$
F3	as F2, except g is replaced by the one from F1	$[0,1]$
F4	as F2, except x_i is replaced by x_i^α, where $i \in \{1,\cdots,m(t)-1\}, \alpha > 0$	$[0,1]$
F5	as F2, except $g = \sum_{i=m(t)}^{n}(x_i - G(\bar{t}))^2$ where $G(\bar{t}) = \lvert\sin(0.5\pi\bar{t})\rvert$, $\bar{t} = \frac{1}{n_{\bar{t}}}\lfloor\frac{\tau}{\tau_{\bar{t}}}\rfloor$	$[0,1]$
F6	$f_1 = (1+g)0.5\prod_{i=1}^{m(t)-1}\cos(x_i^{F(\bar{t})}\pi/2)$ $f_{j=2:m(t)-1} = (1+g)0.5(\prod_{i=1}^{m(t)-j}\cos(x_i^{F(\bar{t})}\pi/2))(\sin(x_{m(t)-j+1}^{F(\bar{t})}\pi/2))$ $f_{m(t)} = (1+g)\sin(x_1^{F(\bar{t})}\pi/2)$ $g = G(\bar{t}) + \sum_{i=m(t)}^{n}(x_i^{F(\bar{t})} - G(\bar{t}))^2$ $G(\bar{t}) = \lvert\sin(0.5\pi\bar{t})\rvert$ $F(\bar{t}) = 1+100\sin^4(0.5\pi\bar{t})$	$[0,1]$

3.3.2 Performance Metrics

In the field of DMO, there is still no standard metric for quantitatively evaluating the performance of algorithms. In this paper, we employ the following two metrics, adapted from two popular metrics used in the literature, for performance assessment. Most commonly used metrics in this field are adapted from the stationary multi-objective optimization. In this paper, we employ the following two popular performance metrics, both of which are able to evaluate the convergence and diversity simultaneously.

- *Mean Inverted Generational Distance (MIGD)* [91]: Let P_t^* is a set of points uniformly sampled along the PF_t, S_t is the set of solutions obtained by an EMO algorithm for approximating the PF_t, and T is a set of discrete time points before changes in a run,

$$\text{MIGD} = \frac{1}{|T|}\sum_{t\in T}\text{IGD}(P_t^*, S_t). \tag{3.10}$$

In particular, as defined by Bosman et al. [61],

$$\mathrm{IGD}(P_t^*, P_t) = \frac{\sum_{\mathbf{x}\in P} dist(\mathbf{x}, P_t^*)}{|S|}, \tag{3.11}$$

where $dist(\mathbf{x}, P_t^*)$ is the Euclidean distance between a point $\mathbf{x} \in S_t$ and its nearest neighbour in P_t^*. Note that the calculation of IGD requires the prior knowledge of the PF. In our experiments, we use the method suggested by Li et al. [75] to sample 10,000 uniformly distributed points on the corresponding PF_t at the t-th time step where $t \in T$.

- *Mean Hypervolume (MHV)*: Let $\mathbf{z}_t^w = (z_1^w, \ldots, z_{m(t)}^w)^T$ be a worst point in the objective space that is dominated by all Pareto-optimal objective vectors at the t-th time step,

$$\mathrm{MHV} = \frac{1}{|T|}\sum_{t\in T} \mathrm{HV}(S_t). \tag{3.12}$$

In particular, as defined by Zitzler et al. [119], HV measures the size of the objective space dominated by solutions in S_t and bounded by $\mathbf{z}_t^w$.

$$\mathrm{HV}(S_t) = \mathrm{VOL}(\bigcup_{\mathbf{x}\in S_t} [f_1(\mathbf{x}), z_1^w] \times \ldots [f_{m(t)}(\mathbf{x}), z_{m(t)}^w]) \tag{3.13}$$

where VOL$(\cdot)$ indicates the Lebesgue measure. Note that solutions, dominated by the worst point, are discarded for HV calculation. For a better presentation, the HV values used in Equation 3.12 are normalized to $[0, 1]$ by dividing $z_t = \prod_{i=1}^m z_i^w$.

3.3.3 EMO Algorithms Used in the Experimental Studies

In our empirical studies, four state-of-the-art EMO algorithms are used for comparisons: the dynamic version of the elitist non-dominated sorting genetic algorithm (DNSGA-II) [9] and MOEA/D with Kalman Filter prediction (MOEA/D-KF) [96]; and their corresponding stationary baseline NSGA-II [32] and MOEA/D [36]. They were chosen because of

their popularity and good performance in both dynamic and static environments. Comparisons with the baseline algorithms are important, because we want to ensure that the dynamic algorithms should at least outperform a stationary algorithm. The following paragraphs provide some brief descriptions of these compared algorithms.

- *DNSGA-II*: To make the classic NSGA-II suitable for handling dynamic optimization problems, [9] suggested to replace some population members with either randomly generated solutions or mutated solutions upon existing ones once a change occurs. As reported in [9], the prior one performs better on DMOPs with severely changing environments while the latter one may work well on DMOPs with moderate changes. In our experiments, we adopt the prior DNSGA-II version in view of its slightly better performance reported in [9].

- *MOEA/D-KF*: This is a recently proposed prediction-based strategy that employs a linear discrete time Kalman Filter to model the movements of the PS in a dynamic environment. Thereafter, this model is used to predict the new location of the PS when a change occurs. Empirical results in [96] has shown that MOEA/D-KF is very competitive for the dynamic optimization and it outperforms the other state-of-the-art predictive strategies, e.g., [91] and [94].

- *NSGA-II*: It at first uses non-dominated sorting to divide the population into several non-domination levels. Solutions in the first several levels have a high priority to be selected as the next parents. The exceeded solutions are trimmed according to the density information.

- *MOEA/D*: This is a representative of the decomposition-based EMO methods. Its basic idea is to decompose the original MOP into several subproblems, either single-objective scalar functions or simplified MOPs. Thereafter, it employs some population-based techniques to solve these subproblems in a collaborative manner.

Each algorithm is independently run 31 times on each problem instance. 300 generations are given to each algorithm before the first change. In other words, the first change

occurs after the first 300 generations. To have a statistically sound conclusion, we use the Wilcoxon rank sum test at the 5% significance level in the comparisons.

The time varying number of objectives $m(t)$ as follows:

$$m(t) = \begin{cases} 3, & t = 1 \\ m(t-1) + 1, & t \in [2, 5] \\ m(t-1) - 1, & t \in [6, 10] \end{cases} \tag{3.14}$$

where $t \in \{1, \cdots, 10\}$ is a discrete time. To investigate the performance of different algorithms under various frequencies of change, τ_t is set as 25, 50, 100, 200, respectively. Table 3.2 and Table 3.3 give the median and the interquartile range (IQR) of the corresponding metric values obtained by different algorithms under various circumstances. In particular, the best metric values are highlighted in the bold face with a grey background. In addition to the metric values, a record of the ranks of the IGD and HV values obtained by different algorithms at each time step will be given. A global rank to each algorithm will be assigned by averaging the ranks obtained at all time steps. The experimental results clearly demonstrate that DTAEA is the best optimizer as it wins on most comparisons and it is always top ranked. In the following paragraphs, we will explain these results in detail.

3.3.4 Results on F1 to F4

F1 to F4 in which only the number of objectives changes with time.

F1 is developed from DTLZ1 [64] which has a multi-modal property to hinder an algorithm from converging to the PF_t, $t \in \{1, \cdots, 10\}$. As shown in Table 3.2 and Table 3.3, the performance of DTAEA is robust as it obtains the best MIGD and MHV values under all four frequencies of change. Figure 3.6(a) shows the trajectories of IGD values obtained by different algorithms across the whole evolution process; while Figure 3.9(a) shows the average ranks of IGD obtained by different algorithms at each time step.

From these two figures, we clearly see that DTAEA shows the best performance at every time step. It is worth noting that the performance of DTAEA has some fluctuations when $\tau_t = 25$. This is because, under a high frequency of change, DTAEA can hardly drive the solutions fully converge to the PF_t before the environment changes. However, with the decrease of the frequency of change, i.e., the increase of τ_t, the performance of DTAEA becomes stable. As for the other four algorithms, the performance of different algorithms fluctuates a lot at different time steps. It is interesting to note that the performance of the stationary algorithms, i.e., NSGA-II and MOEA/D, are comparable to their dynamic counterparts, i.e., DNSGA-II and MOEA/D-KF. In particular, under a high frequency of change, i.e, $\tau_t = 25$, NSGA-II and MOEA/D have shown a better performance than DNSGA-II and MOEA/D-KF at every time step. This implies that the dynamic handling techniques of DNSGA-II and MOEA/D-KF might not be capable of handling the expansion or contraction of the objective space. Even worse, these mechanisms can have an adverse effect to the population for adapting to the changing environments. In addition, we also notice that NSGA-II and DNSGA-II show better performance than MOEA/D and MOEA/D-KF at the first several time steps; whereas their performance degenerates significantly afterwards. This is because NSGA-II can have a faster convergence than MOEA/D when the number of objectives is relatively small. The increase of the number of objectives does not influence the population convergence. Thus, the competitive IGD values obtained at the 2- and 3-objective cases have an accumulative effect which makes NSGA-II and DNSGA-II maintain a competitive performance at the first couple of time steps. However, the ineffectiveness of NSGA-II and DNSGA-II for handling problems with a large number of objectives [120] leads to their poor performance at the latter time steps. Even worse, their poor performance at 6- and 7-objective scenarios also disseminate a negative accumulation which results in their poor performance when decreasing the number of objectives.

F2 is developed from DTLZ2 [64] which is a relatively simple benchmark problem. From Figure 3.6(b) and Figure 3.9(b), we also find that DTAEA has shown the consistently

best performance across all time steps. It is interesting to note that the MIGD and MHV values obtained by all five algorithms lie in the same scale. This is because F2 does not pose too much challenge to the algorithms for converging to the PF_t. Thus all algorithms are able to adapt their populations to the changing environments within a reasonable number of function evaluations. Due to this reason, as shown in Figure 3.6(b), the IGD trajectories of NSGA-II and DNSGA-II surge up to a relatively high level when the number of objectives becomes large; whereas their IGD trajectories gradually go down when decreasing the number of objectives. Moreover, similar to the observations on F1, DNSGA-II and MOEA/D-KF do not show superior performance than their corresponding stationary counterparts on F2.

F3 is developed from DTLZ3 [64] which has the same PF_t, $t \in \{1, \cdots, 10\}$, as F2 but has a multi-modal property. Due to the multi-modal property, which hinders the population from approaching the PF_t, the IGD trajectories of NSGA-II and DNSGA-II can hardly drop down when decreasing the number of objectives. In the meanwhile, we still notice that the dynamic handling techniques of DNGSA-II and MOEA/D-KF do not help the population adapt to the changing environments. Even worse, the injected solutions provide a false information about the newly changed PF/PS, which is harmful to the evolution process.

F4 is developed from DTLZ4 [64] which also has the same PF_t, $t \in \{1, \cdots, 10\}$, as F2 but has a parametric mapping that introduces a biased density of Pareto-optimal solutions towards some particular coordinates. In this case, F4 not only poses significant challenges for handling the changing number of objectives, but also requires that the algorithm can have a good balance between convergence and diversity. The superior performance of DTAEA can be attributed to the dedicated operations of two co-evolving populations for balancing convergence and diversity during the whole evolution process. Similar to observations on the previous benchmark problems, the stationary algorithms still have shown competitive performance than their dynamic counterparts at most time steps.

3.3.5 Results on F5 and F6

Different from F1 to F4, F5 and F6 have a time varying PS. Here we use the Equation 3.14 to define the time varying number of objectives; as for the parameters related to the time varying PS, we set its change frequency as $\frac{1}{\tau_t} = 0.2$ and change severity [12] as $n_t = 10$ (these settings are widely used in the literature, e.g., [101, 121] and [122]). From the experimental results shown in Table 3.2 and Table 3.3, we find that DTAEA has shown the best performance on almost all comparisons (32 out of 32 for MIGD and 30 out of 32 for MHV). In the following paragraphs, we will explain these results in detail.

F5 is developed from F2 and is with a time varying PS of which the position is changed with time. Although the overall performance of DTAEA is the best as shown in Table 3.2 and Table 3.3, its performance fluctuates significantly under a high frequency of change, i.e., $\tau_t = 25$ and $\tau_t = 50$, as shown in Figure 3.8(a) and Figure 3.11(a). It is worth noting that DTAEA actually does not react towards the time varying PS. Thus, under a high frequency of change, DTAEA does not make a good adaptation to two kinds of dynamics. However, as discussed in Section 3.3.4, F2 does not pose too much difficulty to the algorithm for converging to the PF. This explains the comparable performance achieved by the other four algorithms under a high frequency of change. Nevertheless, even without any specific dynamic handling technique for the time varying PS, the complementary effects of two archives of DTAEA help the population adapt to the changing environments when decreasing the frequency of change, i.e., increasing τ_t. Furthermore, NSGA-II and MOEA/D have still shown comparable performance than their dynamic counterparts when encountering two kinds of dynamics. The reason for NSGA-II perform well is: although NSGA-II cannot handle problems with more than three objectives, NSGA-II has the highest convergent speed dealing with two and three objectives. As a result, NSGA-II can achieve a relatively good population in a short time comparing to other algorithms. This helps NSGA-II maintain the advantage even dealing with more than three objectives. The reason for MOEA/D perform well is different: MOEA/D cannot achieve a good population as quick as NSGA-II, however, the diversity maintenance

strategy works well dealing with many-objective situation. This gives a chance to the MOEA/D obtain advantage during that period. F6 is also developed from F2 but it poses more challenges to the algorithm for approaching the PF. Similar to the observations on F5, the performance of DTAEA fluctuate significantly under a high frequency of change as shown in Figure 3.8(b) and Figure 3.11(b). However, since the g function of F6, which controls the difficulty for converging to the PF, is more difficult than that of F5, the superiority of DTAEA is more observable than F5. In addition, although the dynamic handling technique of DNSGA-II and MOEA/D-KF are designed for handling the time varying PS, they do not show better performance than their stationary counterparts. We explain these observations as that the severity of change of the time varying PS is smaller than the expansion or contraction of the PS or PF manifold when changing the number of objectives. Thus, due to the ineffectiveness of the dynamic reaction mechanisms of DNSGA-II and MOEA/D-KF for handling the changing number of objectives, it makes their performance be not much indifferent from their dynamic counterparts.

It is interesting to note that the stationary algorithms, i.e., NSGA-II and MOEA/D, have shown superior performance to their dynamic counterparts, i.e., DNSGA-II and MOEA/D-KF, under a high frequency of change, i.e., $\tau_t = 25$. Furthermore, we also notice that the stationary MOEA/D shows better performance than MOEA/D-KF after the 6-th time step, i.e., when decreasing the number of objectives. In addition, the performance of NSGA-II and DNSGA-II are relatively better than that of MOEA/D and MOEA/D-KF at the first several time steps, i.e., when increasing the number of objectives; whereas their performance becomes poor afterwards. As reported in many studies [120, 123], NSGA-II cannot work well for problems for problems with more than 4 objectives. However, since NSGA-II has a good convergence for the problems with a small number of objectives, its performance is competitive during the first several time steps.

All these observations imply that the dynamic reaction operations of DNSGA-II and MOEA/D-KF might not be capable of handling dynamic problems with a changing number of objectives. Even worse, these dynamic reaction operations can make adverse effects

to the search process when the number of objectives changes.

Table 3.2: Performance Comparisons of DTAEA and the Other Algorithms on MIGD Metric

	τ_t	NSGA-II		DNSGA-II		MOEA/D		MOEA/D-KF		DTAEA	
		MIGD	R	MIGD	R	MIGD	R	MIGD	R	MIGD	R
F1	25	1.40E-2(9.25E-3)†	3.3	1.32E-2(6.51E-3)†	3.3	1.53E-3(2.18E-4)†	3.1	2.05E-2(1.81E-2)†	4.2	**5.50E-4(1.99E-4)**	**1.1**
	50	3.36E-2(3.33E-2)†	3.7	3.38E-2(2.07E-2)†	3.8	8.53E-4(9.43E-4)†	2.9	5.22E-3(6.23E-3)†	3.5	**3.88E-4(1.03E-5)**	**1**
	100	1.99E-2(5.68E-2)†	3.9	2.84E-2(6.27E-2)†	4.1	7.36E-4(4.29E-4)†	2.7	9.78E-4(6.03E-4)†	3.3	**3.64E-4(2.49E-6)**	**1**
	200	1.71E-2(5.38E-2)†	4.3	7.83E-3(3.45E-2)†	4.1	6.47E-4(8.72E-4)†	2.6	6.75E-4(4.58E-4)†	3	**3.55E-4(1.41E-6)**	**1**
F2	25	1.92E-3(1.27E-4)†	3.2	2.02E-3(2.12E-4)†	3.6	2.19E-3(1.17E-4)†	3.3	2.26E-3(6.71E-5)†	3.8	**1.26E-3(4.58E-6)**	**1.1**
	50	2.20E-3(9.07E-5)†	3.6	2.24E-3(1.47E-4)†	3.7	2.08E-3(5.81E-5)†	3.2	2.10E-3(6.03E-5)†	3.5	**1.25E-3(3.26E-6)**	**1**
	100	2.45E-3(1.37E-4)†	3.8	2.47E-3(1.41E-4)†	3.9	2.01E-3(2.89E-5)†	3.1	1.99E-3(3.51E-5)†	3.1	**1.22E-3(1.30E-6)**	**1**
	200	2.46E-3(1.61E-4)†	3.8	2.50E-3(1.48E-4)†	3.9	1.93E-3(4.26E-5)†	3.1	1.92E-3(3.41E-5)†	3.1	**1.20E-3(2.20E-6)**	**1**
F3	25	2.15E-2(1.78E-2)†	3.1	2.62E-2(1.82E-2)†	3.1	4.68E-3(1.25E-2)†	3.5	4.90E-2(6.92E-2)†	4.1	**2.06E-3(1.34E-3)**	**1.1**
	50	9.09E-2(3.82E-2)†	3.7	8.88E-2(4.13E-2)†	3.9	2.96E-3(5.33E-4)†	2.9	1.02E-2(5.72E-3)†	3.4	**1.39E-3(1.06E-4)**	**1.1**
	100	1.61E-1(1.08E-1)†	4	1.35E-1(1.22E-1)†	4	2.22E-3(1.31E-4)†	2.7	2.57E-3(9.73E-4)†	3.2	**1.24E-3(9.04E-6)**	**1**
	200	1.82E-1(2.01E-1)†	4.2	1.91E-1(1.47E-1)†	4.2	2.09E-3(4.00E-3)†	2.6	2.16E-3(9.26E-4)†	2.9	**1.22E-3(4.17E-6)**	**1**
F4	25	3.06E-3(5.08E-4)†	2.7	3.66E-3(5.08E-4)†	3.4	4.19E-3(3.56E-4)†	4.1	4.02E-3(2.12E-4)†	3.8	**1.32E-3(7.80E-5)**	**1.1**
	50	2.93E-3(3.78E-4)†	3.1	3.19E-3(3.71E-4)†	3.4	3.74E-3(2.07E-4)†	3.9	3.62E-3(3.94E-4)†	3.6	**1.24E-3(4.05E-6)**	**1**
	100	3.19E-3(3.12E-4)†	3.4	3.16E-3(4.91E-4)†	3.4	3.38E-3(6.86E-4)†	3.6	3.15E-3(3.86E-4)†	3.6	**1.22E-3(1.50E-6)**	**1**
	200	2.79E-3(1.82E-4)†	3.3	2.79E-3(3.63E-4)†	3.3	2.73E-3(4.51E-4)†	3.6	2.78E-3(5.72E-4)†	3.8	**1.20E-3(1.35E-6)**	**1**
F5	25	7.93E-3(3.37E-3)†	4.2	3.69E-3(2.93E-4)†	4	2.61E-3(2.62E-4)	2.6	**2.56E-3(1.78E-4)**‡	2.5	2.61E-3(9.71E-5)	**1.8**
	50	3.36E-3(2.45E-4)†	3.8	3.69E-3(2.36E-4)†	4.5	2.31E-3(1.25E-4)†	2.6	2.24E-3(1.08E-4)†	2.5	**1.91E-3(6.69E-5)**	**1.7**
	100	2.02E-3(1.80E-4)†	3.2	2.24E-3(1.23E-4)†	4	2.13E-3(9.99E-5)†	3.5	1.99E-3(4.51E-5)†	2.9	**1.45E-3(2.98E-5)**	**1.3**
	200	1.91E-3(1.06E-4)†	2.9	2.20E-3(1.23E-4)†	4.3	2.02E-3(7.70E-5)†	3.4	1.93E-3(4.98E-5)†	3.1	**1.36E-3(6.77E-6)**	**1.2**
F6	25	8.10E-3(1.09E-3)†	4.3	5.19E-3(7.90E-4)†	3.7	3.12E-3(3.53E-4)†	2.7	**2.93E-3(1.86E-4)**	2.5	2.98E-3(1.76E-4)	**1.7**
	50	5.27E-3(5.91E-4)†	4.5	5.60E-3(4.67E-4)†	4.1	2.68E-3(7.39E-5)†	2.6	2.71E-3(1.72E-4)†	2.5	**2.08E-3(5.08E-5)**	**1.3**
	100	3.56E-3(3.48E-4)†	4.2	3.29E-3(1.11E-4)†	4.2	2.36E-3(1.95E-4)†	2.8	2.32E-3(1.08E-4)†	2.8	**1.56E-3(2.25E-5)**	**1.1**
	200	3.90E-3(3.72E-4)†	4.4	3.10E-3(1.46E-4)†	4	2.24E-3(9.08E-5)†	2.6	2.16E-3(1.06E-4)†	2.8	**1.46E-3(2.09E-5)**	**1.2**

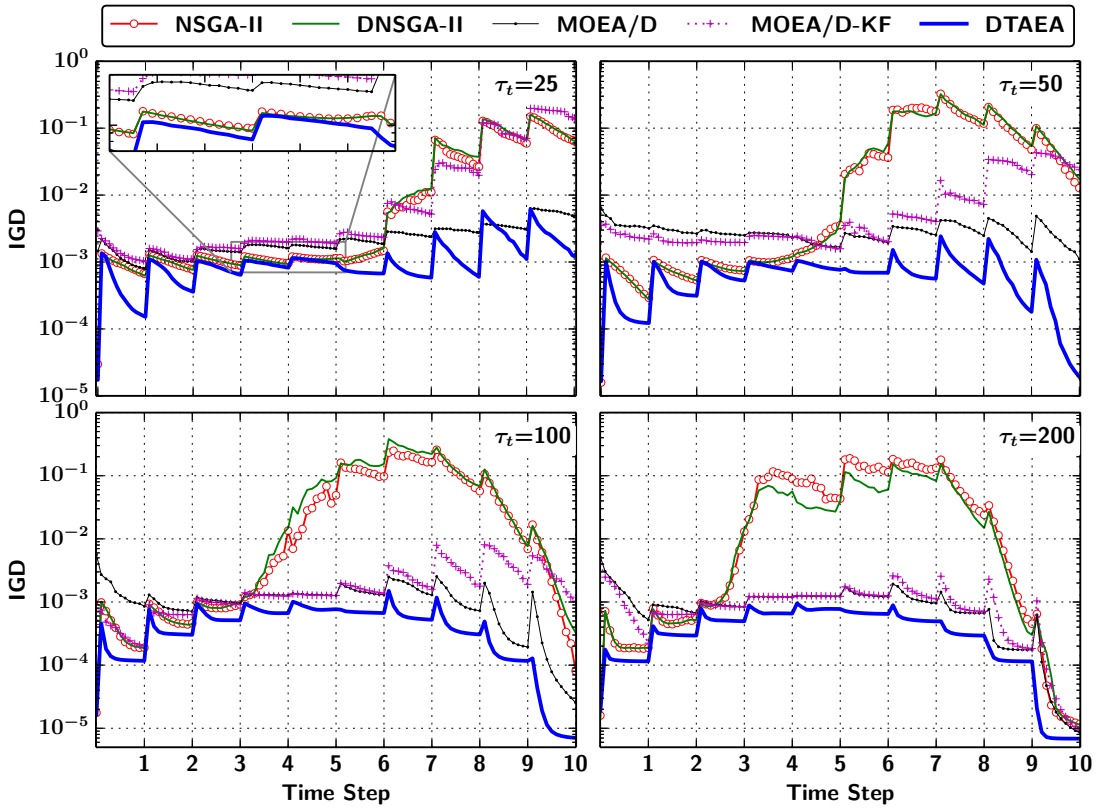
R denotes the global rank assigned to each algorithm by averaging the ranks obtained at all time steps. Wilcoxon's rank sum test at a 0.05 significance level is performed between DTAEA and each of NSGA-II, DNSGA-II, MOEA/D and MOEA/D-KF. † and ‡ denote the performance of the corresponding algorithm is significantly worse than and better than that of DTAEA, respectively. The best median value is highlighted in boldface with grey background.

3.4 Further Analysis

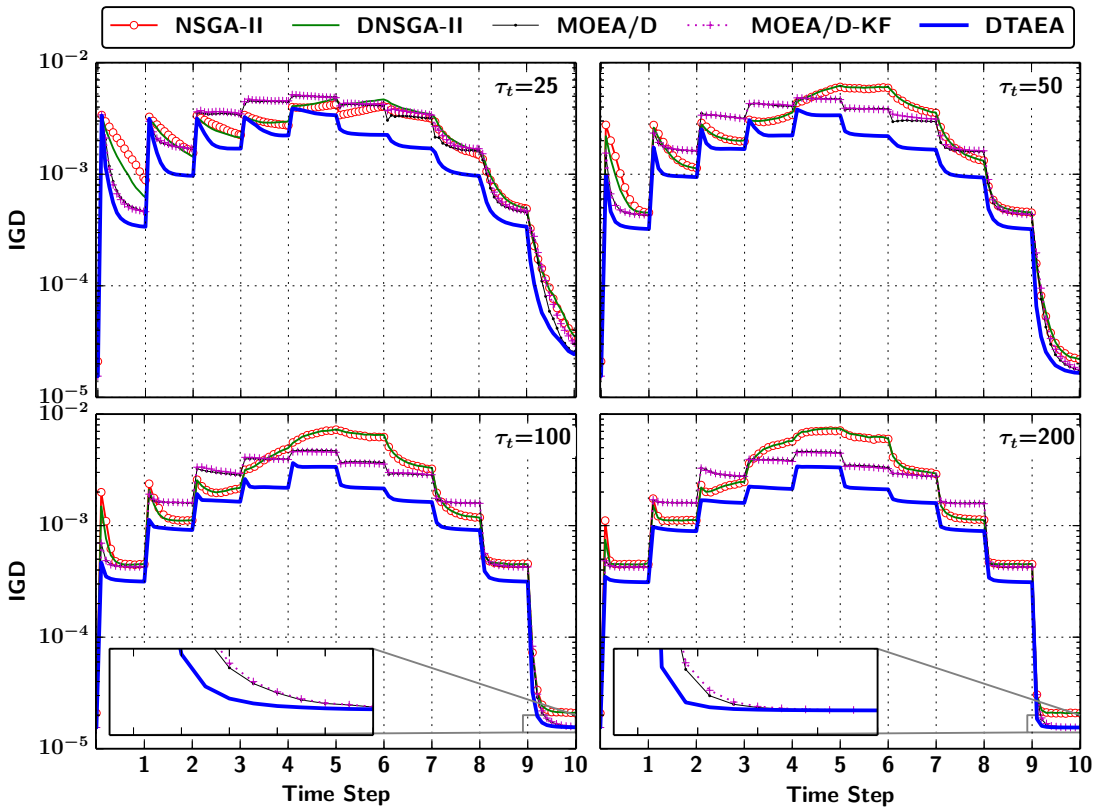
From the experimental studies in Section 3.3.4 and Section 3.3.5, it is clear to see the superiority of the proposed DTAEA for handling MOPs with a dynamically changing number of objectives. In this section, further research questions will be discussed with extra experiment and analysis.

3.4.1 Research Question 1: Effects of the Reconstruction Mechanism

As discussed above, the population convergence might not be affected when increasing the number of objectives. However, the diversity enhancement strategy, e.g., the random

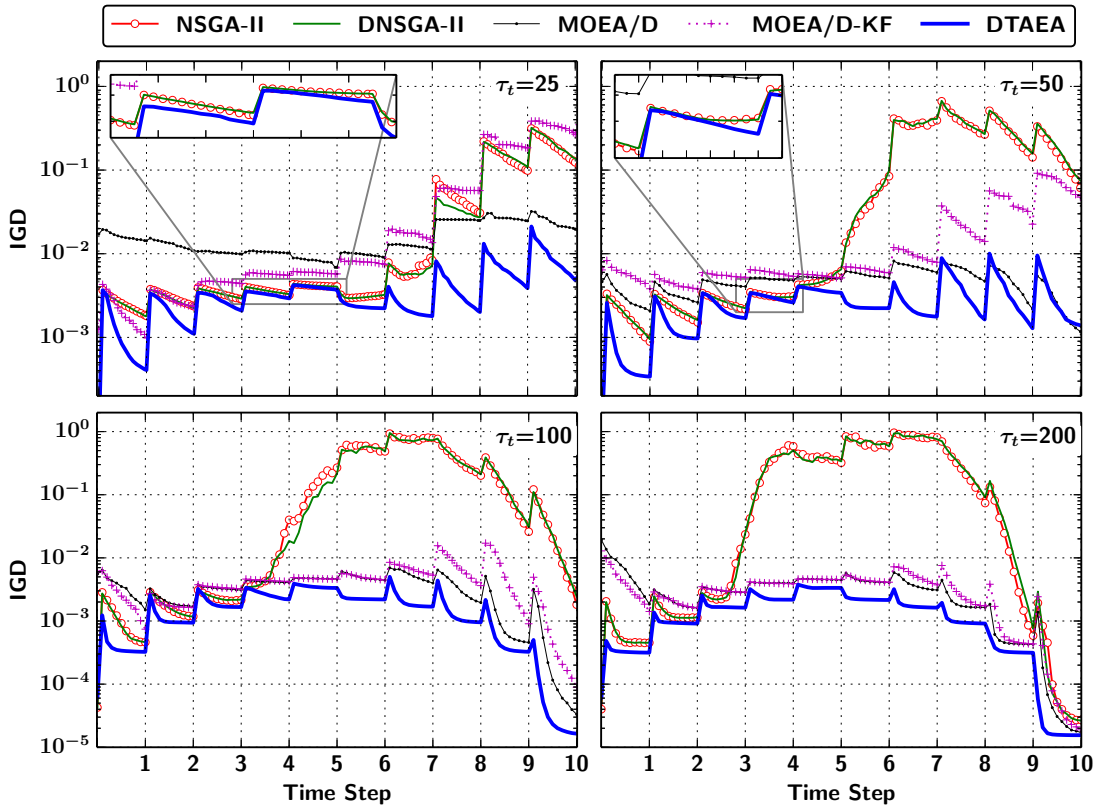


(a) F1

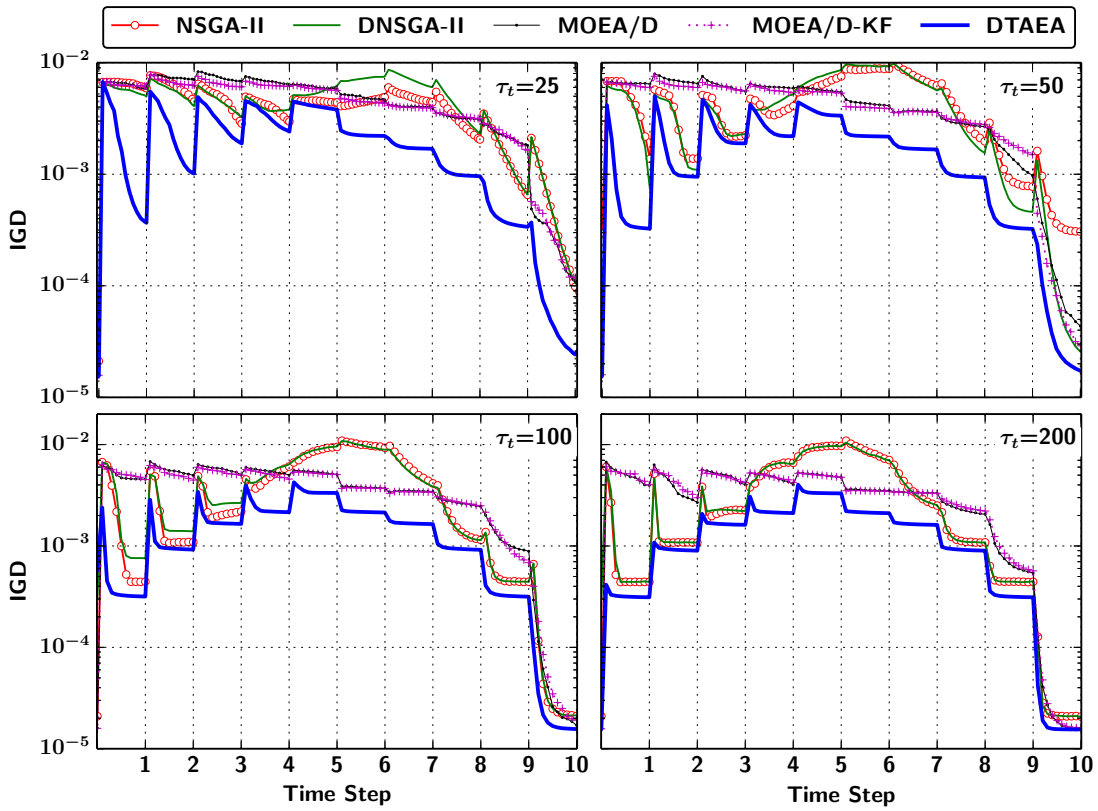


(b) F2

Figure 3.6: IGD trajectories across the whole evolution process(F1,F2).

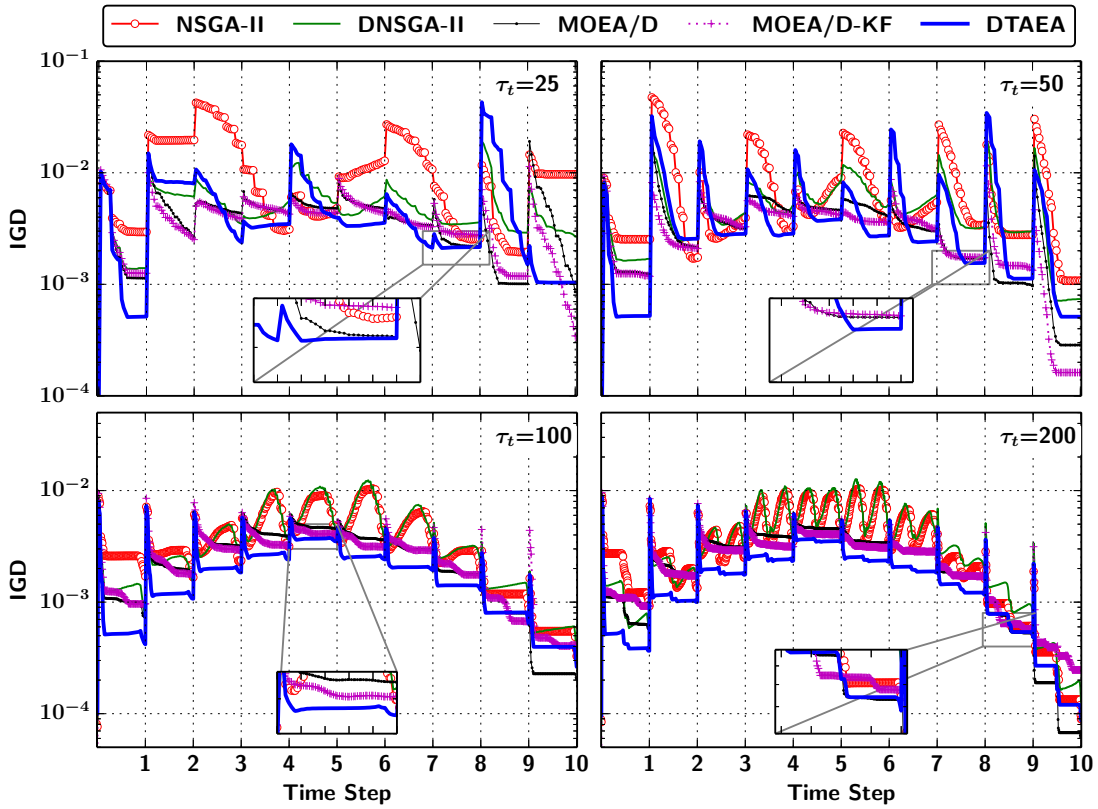


(a) F3

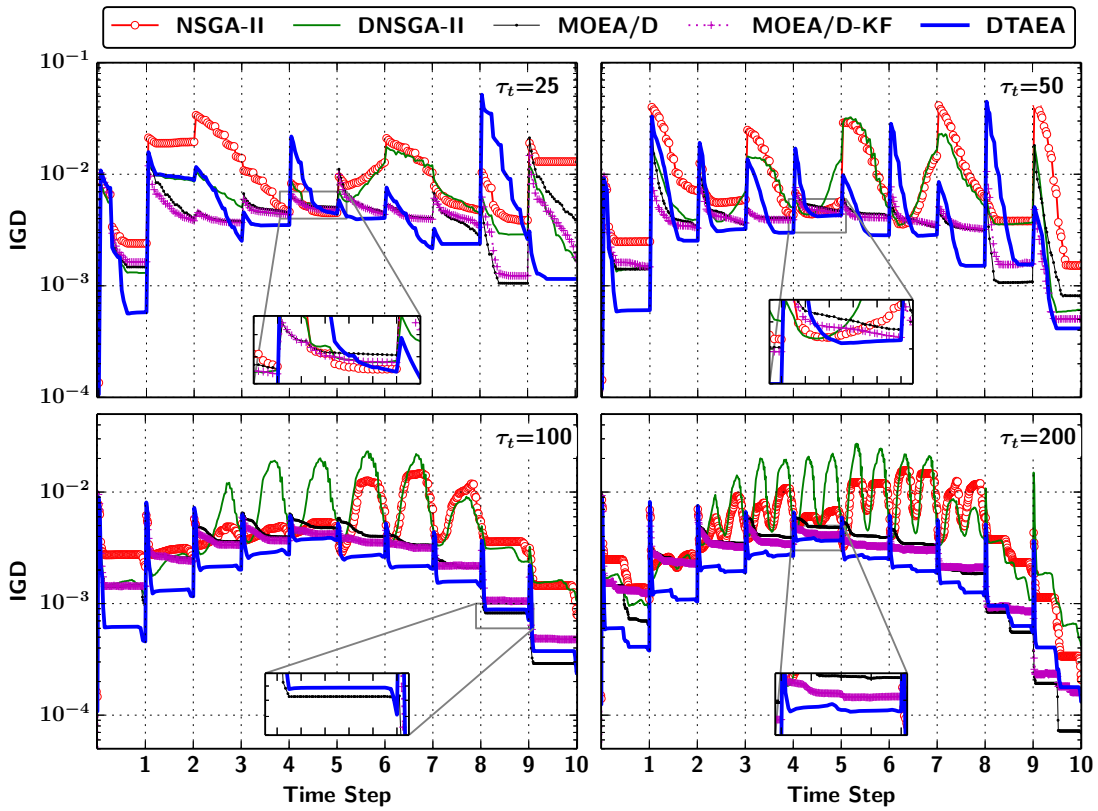


(b) F4

Figure 3.7: IGD trajectories across the whole evolution process(F3,F4).

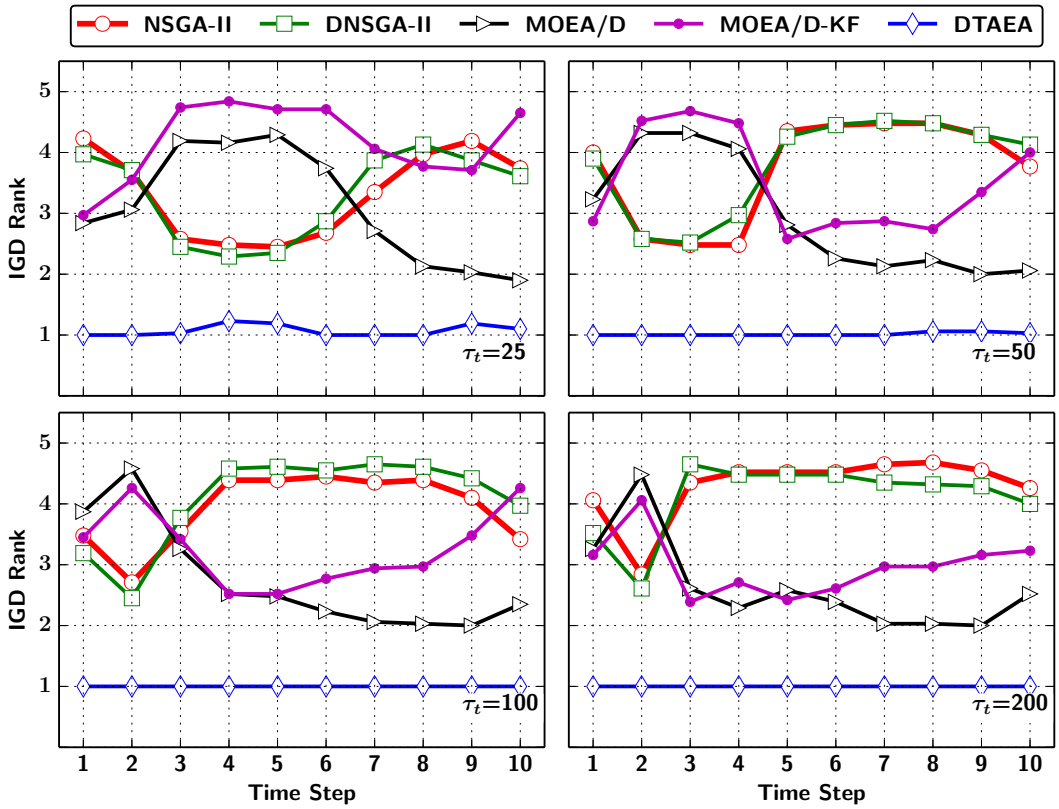


(a) F5

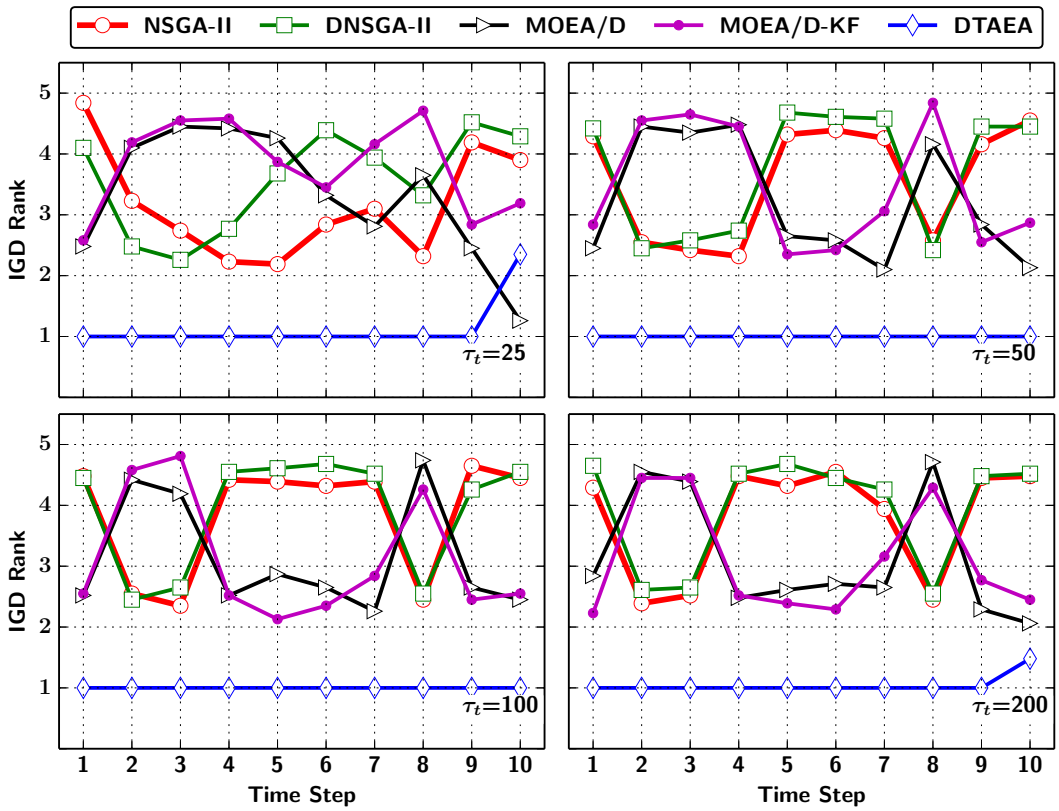


(b) F6

Figure 3.8: IGD trajectories across the whole evolution process(F5,F6).

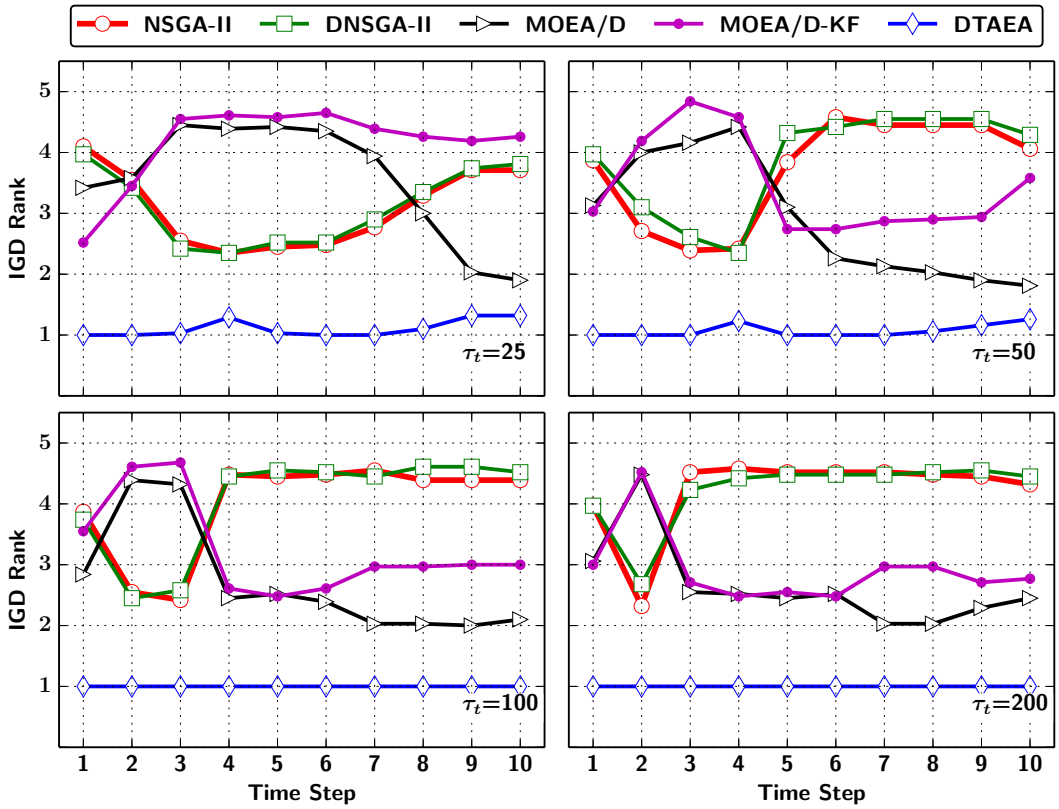


(a) F1

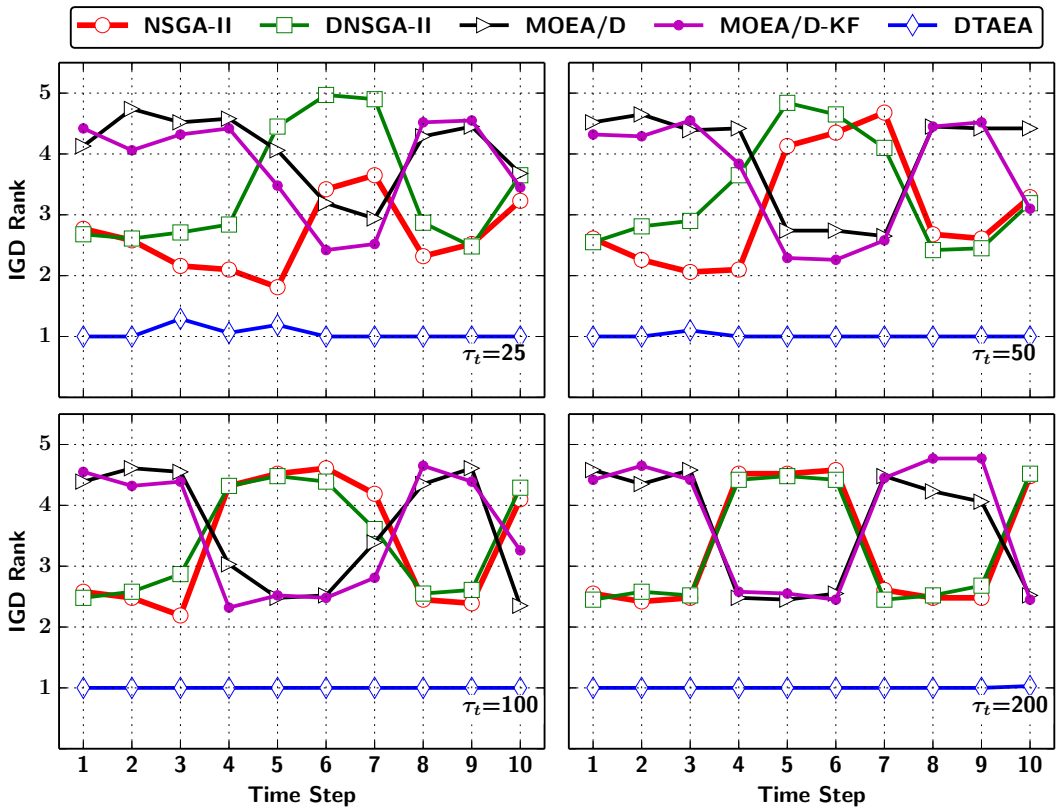


(b) F2

Figure 3.9: The rank of IGD obtained by different algorithms at each time step(F1,F2).

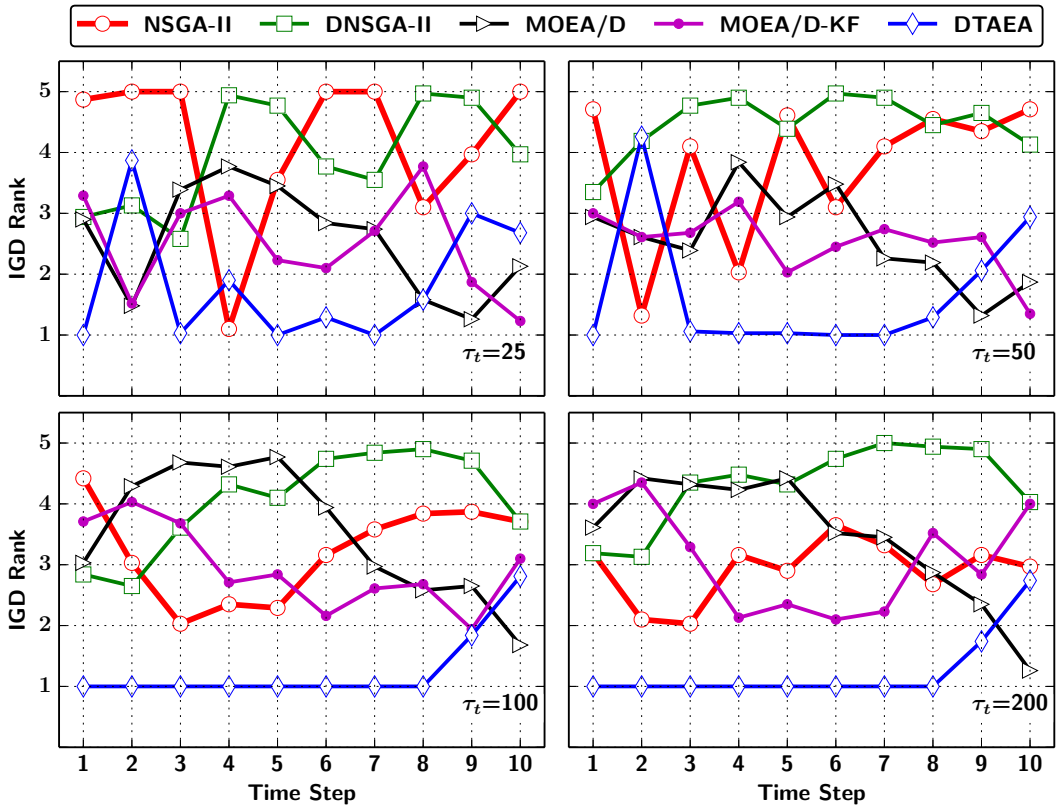


(a) F3

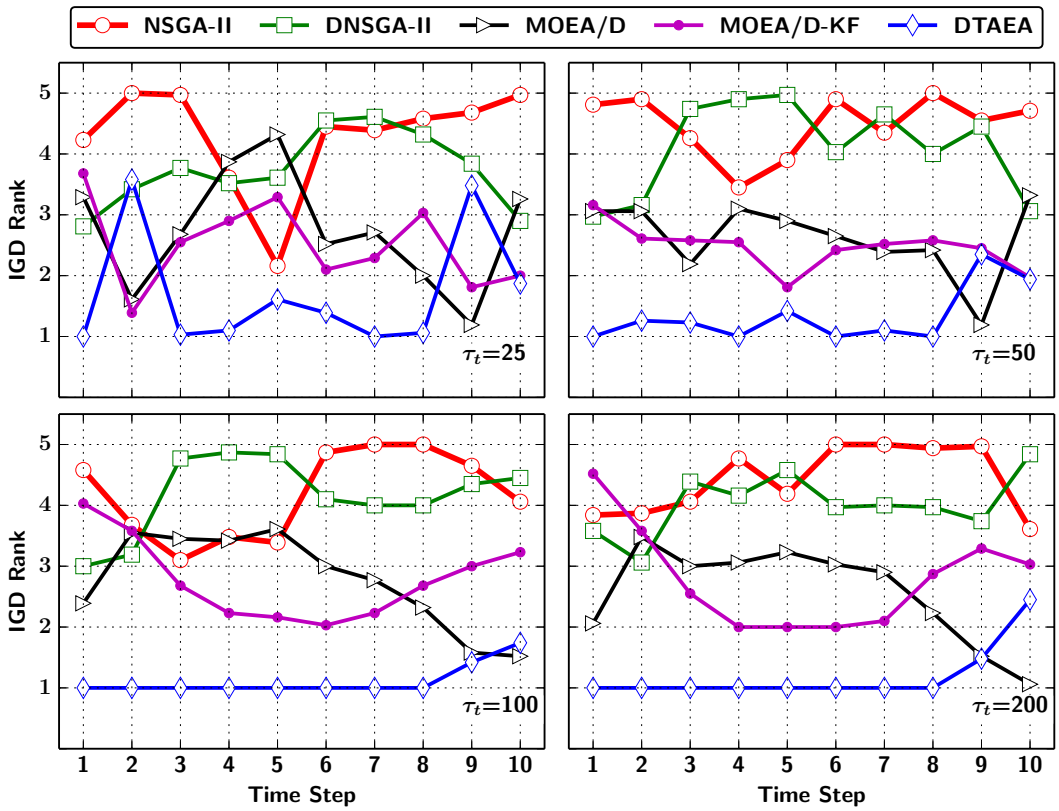


(b) F4

Figure 3.10: The rank of IGD obtained by different algorithms at each time step(F3,F4).



(a) F5



(b) F6

Figure 3.11: The rank of IGD obtained by different algorithms at each time step(F5,F6).

Table 3.3: Performance Comparisons of DTAEA and the Other Algorithms on MHV Metric

	τ_t	NSGA-II		DNSGA-II		MOEA/D		MOEA/D-KF		DTAEA	
		HV	R	HV	R	HV	R	HV	R	HV	R
F1	25	74.8%(9.58E-2)†	3.6	72.3%(8.08E-2)†	3.7	97.3%(1.09E-2)†	2.7	74.6%(4.33E-2)†	3.9	**99.7%(6.03E-3)**	**1.1**
	50	60.1%(1.16E-1)†	4.1	61.2%(1.02E-1)†	4.1	99.7%(3.40E-2)†	2.6	78.0%(1.22E-1)†	3.2	**100.0%(8.39E-5)**	**1**
	100	74.9%(2.10E-1)†	4	69.3%(2.43E-1)†	4.1	99.9%(1.65E-2)†	2.7	99.0%(2.07E-2)†	3.2	**100.0%(1.84E-5)**	**1**
	200	76.0%(3.43E-1)†	4.1	83.3%(3.58E-1)†	3.9	100.0%(4.65E-2)†	2.8	100.0%(2.15E-2)†	3.2	**100.0%(2.69E-5)**	**1**
F2	25	92.4%(3.47E-3)†	4.1	92.4%(6.01E-3)†	4	93.5%(2.91E-3)†	2.6	93.4%(1.36E-3)†	3.1	**94.4%(6.26E-5)**	**1**
	50	91.9%(4.14E-3)†	4.2	91.7%(5.28E-3)†	4.1	93.8%(1.06E-3)†	2.6	93.7%(1.36E-3)†	3.1	**94.4%(3.09E-5)**	**1**
	100	91.0%(5.33E-3)†	4.3	90.6%(8.46E-3)†	4.4	93.9%(4.55E-4)†	2.5	93.9%(6.22E-4)†	2.8	**94.5%(8.88E-6)**	**1**
	200	90.8%(6.80E-3)†	4.3	90.4%(7.10E-3)†	4.4	94.0%(5.10E-4)†	2.6	94.0%(5.09E-4)†	2.7	**94.5%(4.03E-6)**	**1**
F3	25	67.1%(7.75E-2)†	3.4	69.3%(7.98E-2)†	3.3	80.6%(6.04E-1)†	3.4	59.9%(5.09E-2)†	3.8	**90.6%(9.47E-2)**	**1.1**
	50	49.9%(2.21E-2)†	4	49.3%(3.46E-2)†	4.1	89.8%(2.96E-2)†	2.5	67.2%(5.19E-2)†	3.3	**94.0%(3.66E-3)**	**1.1**
	100	45.9%(6.49E-2)†	4.3	44.8%(7.63E-2)†	4.3	93.2%(8.02E-3)†	2.4	91.3%(6.33E-2)†	3	**94.4%(3.32E-4)**	**1**
	200	44.0%(7.13E-2)†	4.2	44.6%(7.75E-2)†	4.2	93.6%(7.25E-2)†	2.7	93.3%(5.40E-2)†	3	**94.4%(1.26E-4)**	**1**
F4	25	91.2%(4.96E-3)†	3.3	88.8%(2.09E-2)†	3.9	90.8%(1.67E-2)†	3.4	91.7%(8.38E-3)†	3.3	**94.4%(1.60E-4)**	**1**
	50	88.8%(2.04E-2)†	3.6	87.6%(2.81E-2)†	3.7	91.9%(9.97E-3)†	3.4	92.1%(1.41E-2)†	3.3	**94.4%(1.16E-5)**	**1**
	100	86.6%(2.83E-2)†	3.6	86.7%(3.45E-2)†	3.5	92.4%(1.31E-2)†	3.4	92.6%(8.25E-3)†	3.5	**94.5%(3.26E-6)**	**1**
	200	88.8%(1.32E-2)†	4	88.5%(2.65E-2)†	3.9	92.9%(1.13E-2)†	3	92.9%(1.26E-2)†	3.2	**94.5%(4.19E-6)**	**1**
F5	25	58.6%(1.38E-1)†	4.5	87.0%(2.36E-2)†	4.1	93.3%(6.10E-3)†	2.2	**93.3%(2.82E-3)**‡	2.3	89.4%(6.64E-3)	**2**
	50	86.3%(1.60E-2)†	4.3	85.9%(1.60E-2)†	4.3	94.0%(2.02E-3)†	2.1	**94.1%(3.35E-3)**‡	**1.9**	93.7%(1.25E-3)	2.4
	100	93.4%(5.24E-3)†	3.8	93.5%(4.20E-3)†	4.3	94.3%(1.15E-3)†	2.9	94.4%(1.95E-3)†	2.3	**94.6%(3.62E-4)**	**1.7**
	200	94.0%(3.40E-3)†	3.7	93.3%(4.36E-3)†	4.5	94.5%(6.96E-4)†	2.9	94.5%(2.20E-3)†	2.3	**94.8%(1.84E-4)**	**1.6**
F6	25	60.1%(6.41E-2)†	4.7	75.8%(6.36E-2)†	3.8	93.2%(1.72E-2)†	2.2	**93.5%(2.77E-3)**‡	**2.1**	89.1%(9.72E-3)	2.3
	50	78.9%(3.83E-2)†	4.6	73.6%(2.76E-2)†	4.3	94.0%(1.71E-3)†	1.9	**93.9%(1.73E-3)**‡	**1.8**	93.7%(1.45E-3)	2.4
	100	88.0%(1.36E-2)†	4.8	91.8%(4.13E-3)†	4	94.3%(1.54E-3)†	2.3	94.2%(2.15E-3)†	2.4	**94.5%(3.50E-4)**	**1.5**
	200	87.9%(1.91E-2)†	4.7	92.1%(5.84E-3)†	4.1	94.5%(5.71E-4)†	2.5	94.5%(1.35E-3)†	2.3	**94.7%(3.39E-4)**	**1.4**

R denotes the global rank assigned to each algorithm by averaging the ranks obtained at all time steps. Wilcoxon's rank sum test at a 0.05 significance level is performed between DTAEA and each of NSGA-II, DNSGA-II, MOEA/D and MOEA/D-KF. † and ‡ denote the performance of the corresponding algorithm is significantly worse than and better than that of DTAEA, respectively. The best median value is highlighted in boldface with grey background.

injection of DNSGA-II, has an adverse effect to the population convergence. In the meanwhile, it cannot provide enough diversified information to remedy the loss of population diversity. As for the prediction strategy, e.g., the Kalman filter of MOEA/D-KF, it is usually developed to predict the change of the PS's position or shape, instead of the expansion or contraction of the PS or PF manifold. As a consequence, its large prediction errors are harmful to both the convergence and diversity. Since the stationary EMO algorithms do not make any reaction to the changing environment, the population convergence is not affected; while the self-adaptive property of EA can also help the population gradually adapt to the changed environment. These explain the competitive performance of the stationary EMO algorithms comparing to their dynamic counterparts for solving DMOPs with a dynamically changing number of objectives. In contrast, the CA of our proposed DTAEA, consisted of the elite population, preserves the population convergence; in the meanwhile, the DA of DTAEA, consisted of a set of uniformly sampled solutions from the decision space, provide as much diversified information as possible to help the population

adapt to the changed environment. In Figure 3.12, we keep track of the population distribution for several iterations after the number of objectives increases from two to three. From this figure, we can clearly see that neither NSGA-II nor DNSGA-II is able to spread the population across the expanded PF. As for MOEA/D and MOEA/D-KF, although the population seems to have an acceptable spread along the expanded PF, the solutions are a bit away from the PF. In contrast, our proposed DTAEA successfully spread the solutions across the PF without sacrificing any convergence characteristic.

As discussed in Section 3.1, the spread of the population might not be affected too much when decreasing the number of objectives, while some solutions are propelled away from the PF. In this case, the randomly injected solutions of DNGSA-II cannot provide sufficient selection pressure towards the PF. As for MOEA/D-KF, due to the large prediction errors, the effect of those predicted solutions of the Kalman filter is almost the same as the random solutions. In contrast, DTAEA uses the non-dominated solutions of the elite population to construct the CA; in the meanwhile, it uses the remaining ones, which have a well spread in the objective space, to construct the DA. By these means, the population is able to have a good balance between convergence and diversity in the changed environment. In Figure 3.13, we keep track of the population distribution for several iterations after the number of objectives decreases from four to three. From this figure, we can clearly see that the population spread after decreasing the number of objectives is acceptable in the first place, but having some duplicate solutions. After several iterations, DTAEA achieves a significantly better performance on both convergence and diversity than the other comparative algorithms.

3.4.2 Research Question 2: Effects of the Mating Selection Mechanism

As discussed in Section 3.2.4, the CA and the DA have different characteristics. In particular, the former is concerned more about convergence whereas the latter one is concerned more about the diversity. Their complementary effect, which finally contributes

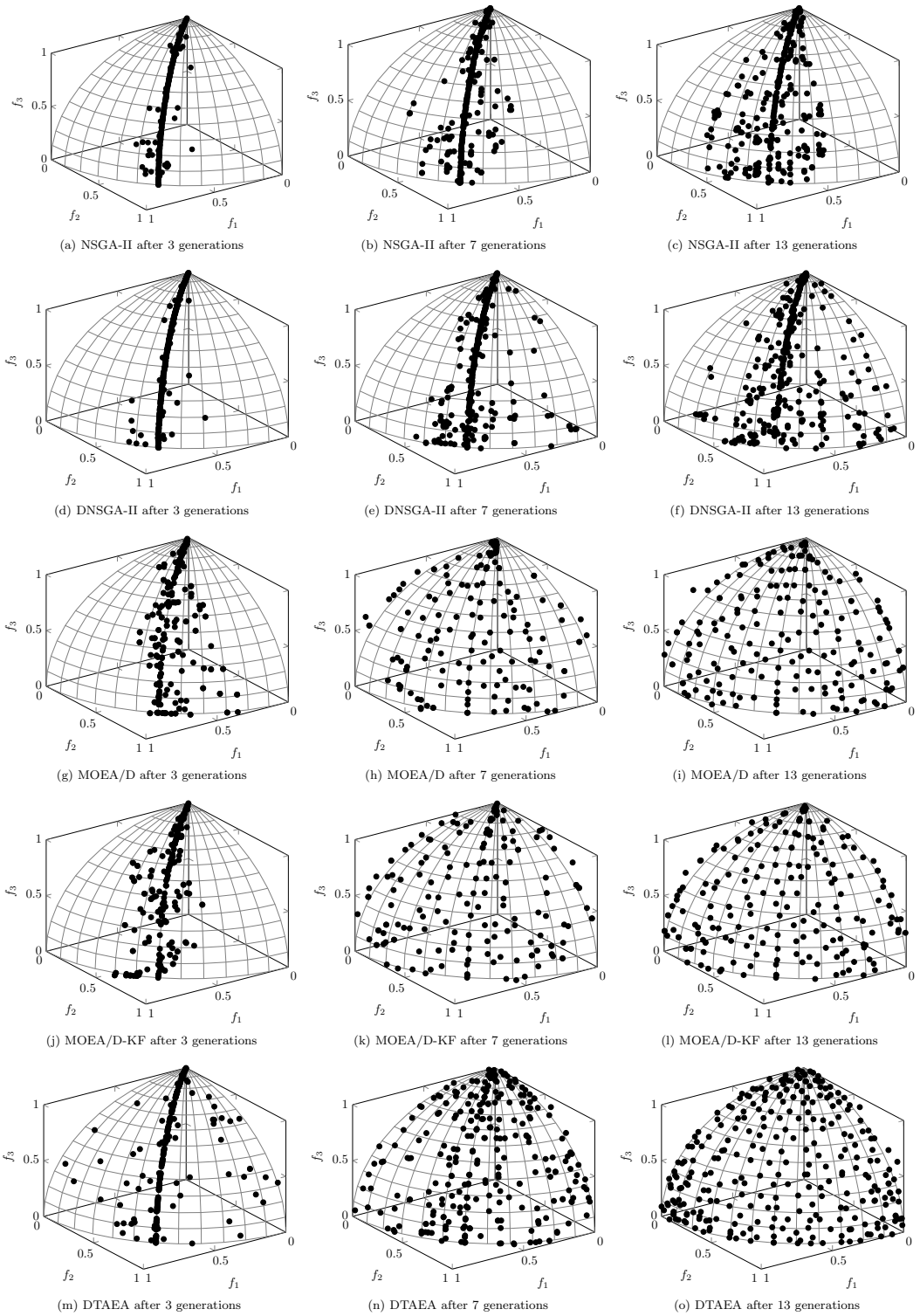
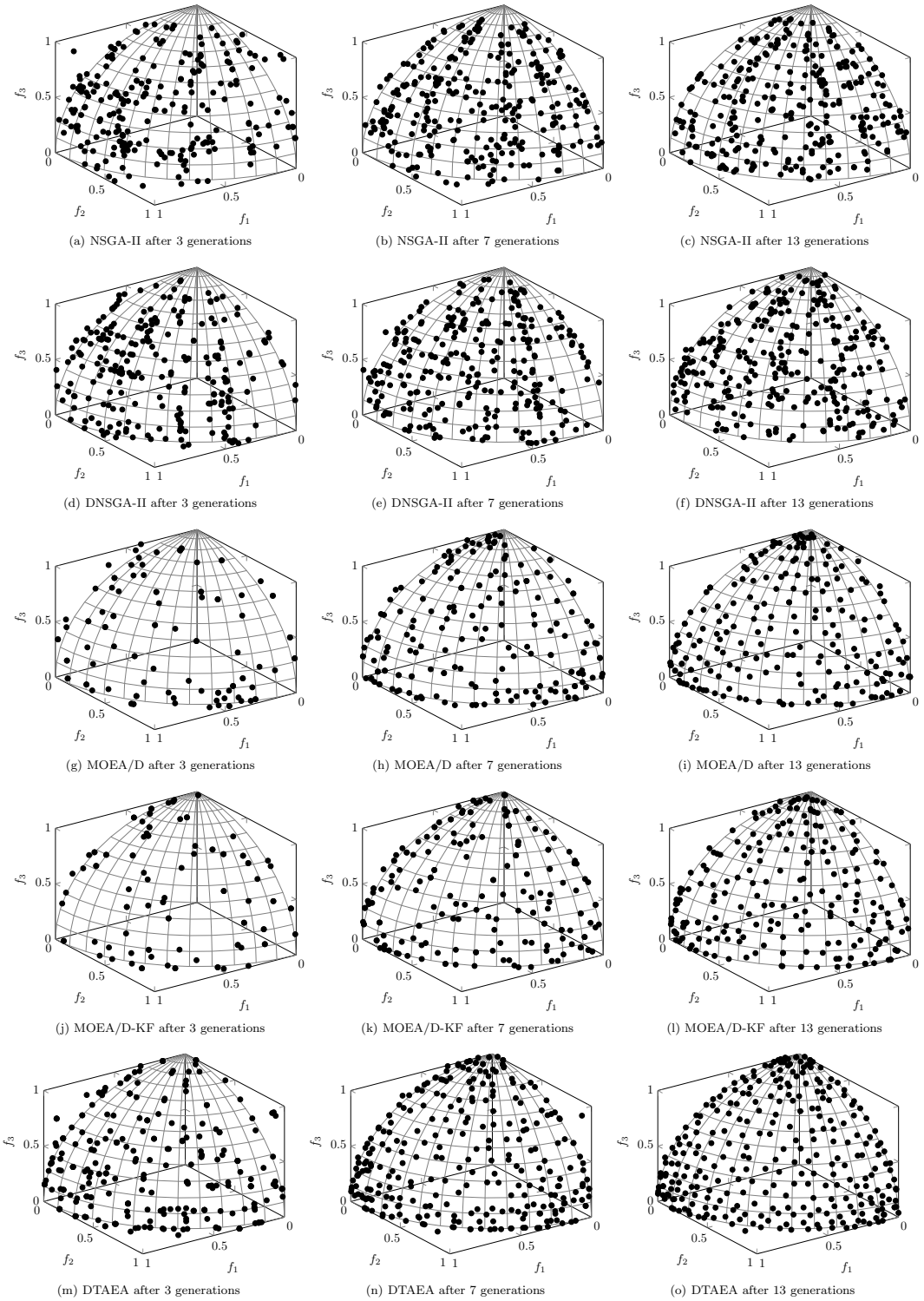


Figure 3.12: Variation of the population distribution when increasing the number of objectives from 2 to 3.



(a) NSGA-II after 3 generations (b) NSGA-II after 7 generations (c) NSGA-II after 13 generations

(d) DNSGA-II after 3 generations (e) DNSGA-II after 7 generations (f) DNSGA-II after 13 generations

(g) MOEA/D after 3 generations (h) MOEA/D after 7 generations (i) MOEA/D after 13 generations

(j) MOEA/D-KF after 3 generations (k) MOEA/D-KF after 7 generations (l) MOEA/D-KF after 13 generations

(m) DTAEA after 3 generations (n) DTAEA after 7 generations (o) DTAEA after 13 generations

Figure 3.13: Variation of the population distribution when decreasing the number of objectives from 4 to 3.

to the balance between convergence and diversity, is implemented by the mating selection mechanism between them. The mating selection mechanism is described in Section 3.2.4. In particular, one of the mating parents is constantly selected from the CA while the selection of the other one depends on the diversity information of the CA. In order to validate the effect of this mating selection mechanism, we choose F2 as the instance where $\tau_t = 50$ and plot the trajectories of the proportion of second mating parent selected from the CA and DA respectively in Figure 3.14. From this figure, we notice that the proportion for selecting the second mating parent from the DA is relatively high at the beginning when the number of objectives is changed. But with the progress of evolution, both mating parents are selected from the CA when the diversity of the CA becomes better.

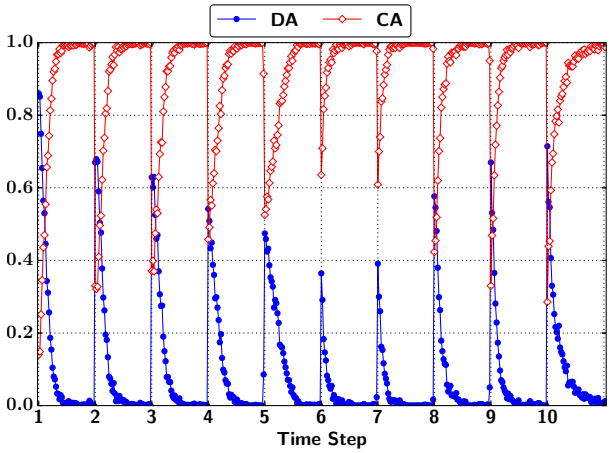


Figure 3.14: Proportion of the second mating parent selected from the CA and DA respectively.

3.4.3 Research Question 3: Effects of the Update Mechanisms

As discussed in Section3.2.3, the update mechanisms are used to maintain the complementary effects between CA and DA. The CA keeps a continuously strong selection pressure for the population convergence; while the DA maintains a set of well diversified solutions. In order to validate the importance of the three different components of DTAEA, we developed three DTAEA variants as follows:

- DTAEA-$v1$: this variant modifies DTAEA by removing the restricted mating selection mechanism proposed in Section 3.2.2. In particular, now the mating parents are respectively selected from the CA and the DA without considering the population distribution.

- DTAEA-$v2$: this variant modifies DTAEA by removing the reconstruction mechanism proposed in Section3.2.2. In other words, it does not make any response to the changing environment.

- DTAEA-$v3$: this variant merely uses the update mechanisms to maintain the CA and the DA whereas it does not make any response to the changing environment. In addition, it does not use the restricted mating selection mechanism as DTAEA-$v1$.

We conduct the experiments on F1 to F6 according to the same experimental settings. Table 3.4 and Table 3.5 give the median and interquartile range (IQR) values according to the MIGD and MHV metrics. From these results, we can see that the original DTAEA, consisted of all three components, has shown clearly better performance than the other three variants. More specifically, as shown in Table 3.4 and Table 3.5, the performance of DTAEA-$v3$ is the worst among all three variants. This observation is reasonable as DTAEA-$v3$ neither responds to the changing environment nor takes advantages of the complementary effect of the CA and the DA for offspring generation. The performance of DTAEA-$v2$ is slightly better than DTAEA-$v3$. Therefore, we can see that even without any response to the changing environment, the restricted mating selection mechanism can also provide some guidance to the search process. As for DTAEA-$v1$, we can see that the performance can be significantly improved by using the reconstruction mechanism proposed in Section 3.2.2 to respond to the changing environment. This superiority is more obvious when the frequency of change is high. By comparing the results between DTAEA-$v1$ and the original DTAEA, we can clearly see the importance of taking advantages of the complementary effect of the CA and the DA for offspring generation. All in all, we can conclude that reconstruction mechanism, update mechanisms and the mating selection

mechanism of DTAEA are of significant importance for handling the DMOPs with a changing number of objectives.

Table 3.4: Performance Comparisons of DTAEA and its Three Variants on MIGD Metric

	τ_t	DTAEA-$v1$		DTAEA-$v2$		DTAEA-$v3$		DTAEA	
		MIGD	R	MIGD	R	MIGD	R	MIGD	R
F1	25	**5.05E-4(1.85E-4)**	**1.9**	7.14E-4(1.72E-4)†	2.8	7.29E-4(2.40E-4)†	2.9	5.50E-4(1.99E-4)	2.3
	50	3.95E-4(2.49E-5)	2.5	4.02E-4(1.54E-5)†	2.6	4.04E-4(1.19E-5)†	2.8	**3.88E-4(1.03E-5)**	**2.1**
	100	3.72E-4(3.97E-6)†	3.5	3.64E-4(3.09E-6)	2	3.65E-4(2.43E-6)	2.7	**3.64E-4(2.49E-6)**	**1.8**
	200	3.59E-4(1.07E-6)†	3.4	3.56E-4(1.20E-6)	1.8	3.59E-4(2.46E-6)†	3.4	**3.55E-4(1.41E-6)**	**1.5**
F2	25	1.28E-3(5.20E-6)†	2.6	1.39E-3(1.97E-5)†	2.8	1.40E-3(4.08E-5)†	3.1	**1.26E-3(4.58E-6)**	**1.5**
	50	1.26E-3(1.59E-6)†	3.4	1.25E-3(1.76E-5)	2	1.25E-3(2.77E-5)†	2.8	**1.25E-3(3.26E-6)**	**1.9**
	100	1.23E-3(1.88E-6)†	3.4	1.22E-3(2.24E-6)†	1.9	1.23E-3(2.32E-6)†	3	**1.22E-3(1.30E-6)**	**1.7**
	200	1.21E-3(1.86E-6)†	3.3	1.20E-3(1.31E-6)†	1.8	1.21E-3(1.73E-6)†	3.2	**1.20E-3(2.20E-6)**	**1.7**
F3	25	**1.92E-3(9.48E-4)**	**1.9**	2.63E-3(1.29E-3)†	3	2.32E-3(8.94E-4)	2.9	2.06E-3(1.34E-3)	2.2
	50	**1.32E-3(4.29E-5)**‡	**2.1**	1.42E-3(5.20E-5)	2.8	1.44E-3(5.95E-5)	2.9	1.39E-3(1.06E-4)	2.3
	100	1.26E-3(5.32E-6)†	3.1	1.25E-3(1.38E-5)	2.1	1.25E-3(1.87E-5)†	2.7	**1.24E-3(9.04E-6)**	**2**
	200	1.23E-3(3.09E-6)†	3	1.22E-3(2.43E-6)†	2.1	1.23E-3(5.26E-6)†	3.2	**1.22E-3(4.17E-6)**	**1.8**
F4	25	**1.29E-3(3.40E-5)**‡	1.7	6.82E-3(2.18E-7)†	3.3	6.82E-3(1.87E-7)†	3.3	1.32E-3(7.80E-5)	**1.7**
	50	1.25E-3(2.17E-6)†	2	6.81E-3(8.96E-8)†	3.3	6.81E-3(7.24E-8)†	3.3	**1.24E-3(4.05E-6)**	**1.4**
	100	1.22E-3(1.58E-6)†	2	6.81E-3(1.93E-7)†	3.3	6.81E-3(1.05E-7)†	3.4	**1.22E-3(1.50E-6)**	**1.4**
	200	1.21E-3(1.85E-6)†	2.1	6.81E-3(1.58E-4)†	3.3	6.81E-3(3.71E-4)†	3.4	**1.20E-3(1.35E-6)**	**1.2**
F5	25	**2.52E-3(9.09E-5)**‡	1.8	7.69E-3(5.69E-4)†	3.3	7.66E-3(3.22E-4)†	3.2	2.61E-3(9.71E-5)	**1.6**
	50	2.02E-3(6.94E-5)†	1.8	1.35E-2(3.26E-4)†	3.5	1.34E-2(2.17E-4)†	3.5	**1.91E-3(6.69E-5)**	**1.2**
	100	1.49E-3(1.72E-5)†	1.7	1.80E-2(4.43E-4)†	3.5	1.79E-2(9.68E-4)†	3.5	**1.45E-3(2.98E-5)**	**1.3**
	200	1.38E-3(1.17E-5)†	1.7	1.95E-2(5.39E-4)†	3.4	1.97E-2(3.73E-4)†	3.6	**1.36E-3(6.77E-6)**	**1.3**
F6	25	**2.90E-3(1.68E-4)**	1.6	1.11E-2(7.24E-4)†	3.5	1.12E-2(5.90E-4)†	3.5	2.98E-3(1.76E-4)	**1.4**
	50	2.22E-3(7.36E-5)†	1.7	1.61E-2(6.16E-4)†	3.5	1.59E-2(7.97E-4)†	3.5	**2.08E-3(5.08E-5)**	**1.3**
	100	1.57E-3(2.31E-5)	1.6	2.10E-2(9.39E-4)†	3.6	2.10E-2(9.48E-4)†	3.4	**1.56E-3(2.25E-5)**	**1.4**
	200	1.46E-3(1.07E-5)	1.6	2.23E-2(7.09E-4)†	3.6	2.19E-2(6.50E-4)†	3.4	**1.46E-3(2.09E-5)**	**1.4**

R denotes the global rank assigned to each algorithm by averaging the ranks obtained at all time steps. Wilcoxon's rank sum test at a 0.05 significance level is performed between DTAEA and each of DTAEA-$v1$, DTAEA-$v2$ and DTAEA-$v3$. † and ‡ denote the performance of the corresponding algorithm is significantly worse than and better than that of DTAEA, respectively. The best median value is highlighted in boldface with grey background.

3.4.4 Research Question 4: Effects under Different Change Frequencies

The experiments on F1 to F4 follow the same experimental settings. However, in order to show the result under different change frequency, τ_t is set as 5, 10, 20, 40, 80, 160, 320, 640. R denotes the global rank assigned to each algorithm by averaging the ranks of MIGD obtained at all time steps. The following Table 3.6 gives the result: For F1, DTAEA becomes the best when $\tau_t \geq 10$. However, the average rank of DTAEA only becomes 1.0 after $\tau_t = 40$. The reason for this result is because F1 is adapted from the DTLZ1 which is hard to achieve convergence. If changes happened for every 5 generations, the DTAEA cannot build an efficient population in such a short period. Noted that all the algorithms' ranks are nearly equal. That is because all the algorithms can hardly

Table 3.5: Performance Comparisons of DTAEA and its Three Variants on MHV Metric

	τ_t	DTAEA-$v1$		DTAEA-$v2$		DTAEA-$v3$		DTAEA	
		HV	R	HV	R	HV	R	HV	R
F1	25	**99.85%(4.60E-3)**	**1.7**	99.57%(3.05E-3)	3	99.45%(6.72E-3)†	3.3	99.7%(6.03E-3)	2
	50	100.00%(1.40E-4)†	2.1	99.98%(1.86E-4)†	2.7	99.98%(2.71E-4)†	3.3	**100.0%(8.39E-5)**	**1.8**
	100	100.00%(1.82E-5)†	2.5	100.00%(2.95E-5)†	2.4	100.00%(2.75E-5)†	3.4	**100.0%(1.84E-5)**	**1.7**
	200	100.00%(1.66E-5)†	2.5	100.00%(2.19E-5)†	2.2	100.00%(2.65E-5)†	3.1	**100.0%(2.69E-5)**	**2.1**
F2	25	94.37%(7.52E-5)†	2.6	93.94%(1.08E-3)†	2.7	93.86%(1.94E-3)†	3.5	**94.4%(6.26E-5)**	**1.2**
	50	94.42%(2.46E-5)†	2.9	94.42%(2.21E-4)†	2.4	94.38%(2.65E-4)†	3.5	**94.4%(3.09E-5)**	**1.2**
	100	94.45%(8.64E-6)†	3.3	94.46%(7.38E-6)†	1.9	94.45%(1.34E-5)†	3.4	**94.5%(8.88E-6)**	**1.5**
	200	94.46%(4.63E-6)†	3.4	94.46%(5.29E-6)†	1.8	94.46%(8.98E-6)†	3.2	**94.5%(4.03E-6)**	**1.7**
F3	25	**91.08%(6.51E-2)**	**1.7**	89.86%(7.13E-2)	3	89.84%(4.89E-2)	3.3	90.6%(9.47E-2)	1.9
	50	**94.24%(2.74E-3)**	**1.9**	93.90%(1.66E-3)†	2.8	93.83%(2.62E-3)†	3.2	94.0%(3.66E-3)	2.1
	100	94.39%(3.78E-4)†	2.5	94.40%(2.96E-4)†	2.4	94.38%(5.41E-4)†	3.3	**94.4%(3.32E-4)**	**1.8**
	200	94.43%(1.60E-4)†	2.9	94.44%(1.52E-4)†	2.1	94.44%(2.72E-4)†	3.2	**94.4%(1.26E-4)**	**1.8**
F4	25	94.39%(1.29E-4)	1.9	79.48%(5.82E-5)†	3.3	79.47%(6.49E-5)†	3.3	**94.4%(1.60E-4)**	**1.5**
	50	94.44%(1.76E-5)†	2.1	79.47%(1.25E-4)†	3.3	79.47%(1.24E-4)†	3.3	**94.4%(1.16E-5)**	**1.3**
	100	94.46%(8.42E-6)†	2.1	79.46%(2.01E-4)†	3.3	79.47%(1.21E-4)†	3.3	**94.5%(3.26E-6)**	**1.2**
	200	94.46%(8.77E-6)†	2	79.47%(1.22E-2)†	3.3	79.47%(3.27E-2)†	3.4	**94.5%(4.19E-6)**	**1.2**
F5	25	**90.37%(5.17E-3)**‡	1.8	65.36%(2.23E-2)†	3.4	64.82%(8.67E-3)†	3.4	89.4%(6.64E-3)	**1.5**
	50	93.53%(1.34E-3)†	1.8	46.58%(1.64E-2)†	3.5	46.18%(2.98E-2)†	3.5	**93.7%(1.25E-3)**	**1.2**
	100	94.53%(4.58E-4)†	1.8	15.51%(3.19E-2)†	3.5	16.54%(3.35E-2)†	3.5	**94.6%(3.62E-4)**	**1.2**
	200	94.73%(3.03E-4)†	1.8	11.56%(1.35E-2)†	3.4	11.09%(6.06E-3)†	3.6	**94.8%(1.84E-4)**	**1.2**
F6	25	**89.31%(1.18E-2)**	1.7	51.61%(4.21E-2)†	3.5	51.54%(2.37E-2)†	3.5	89.1%(9.72E-3)	**1.3**
	50	92.93%(1.19E-3)†	1.8	33.26%(1.84E-2)†	3.5	33.80%(2.49E-2)†	3.5	**93.7%(1.45E-3)**	**1.2**
	100	93.92%(5.32E-4)†	1.8	7.99%(8.41E-3)†	3.5	8.22%(7.57E-3)†	3.5	**94.5%(3.50E-4)**	**1.2**
	200	94.14%(3.41E-4)†	1.8	7.92%(7.58E-3)†	3.5	8.07%(9.59E-3)†	3.5	**94.7%(3.39E-4)**	**1.2**

R denotes the global rank assigned to each algorithm by averaging the ranks obtained at all time steps. Wilcoxon's rank sum test at a 0.05 significance level is performed between DTAEA and each of DTAEA-$v1$, DTAEA-$v2$ and DTAEA-$v3$. † and ‡ denote the performance of the corresponding algorithm is significantly worse than and better than that of DTAEA, respectively. The best median value is highlighted in boldface with grey background.

Table 3.6: Average ranking for different changing frequency

τ_t:		5	10	20	40	80	160	320	640
F1	NSGA-II	3.1	4.1	4.3	4.3	4.0	4.0	4.4	4.3
	DNSGA-II	3.2	3.9	4.3	4.4	4.1	4.3	4.4	4.3
	MOEA/D	**2.9**	2.7	2.8	2.6	2.7	3.0	2.5	2.7
	MOEAD/KF	3.0	3.3	3.1	3.2	3.1	3.0	3.1	3.2
	DTAEA	2.9	**1.4**	**1.2**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
F2	NSGA-II	4.1	4.1	4.2	4.2	4.2	4.3	4.4	4.6
	DNSGA-II	4.1	4.1	4.3	4.4	4.6	4.7	4.9	5.0
	MOEA/D	2.5	2.6	2.7	2.7	2.9	3.0	3.1	3.2
	MOEAD/KF	3.1	3.1	3.1	3.2	3.3	3.3	3.4	3.4
	DTAEA	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
F3	NSGA-II	4.0	4.2	4.3	4.3	4.5	4.6	4.7	4.8
	DNSGA-II	3.6	3.7	3.7	3.9	3.9	4.0	4.1	4.1
	MOEA/D	3.3	3.5	3.7	3.8	3.8	4.0	4.1	4.3
	MOEAD/KF	3.3	3.5	3.5	3.6	3.6	3.6	3.7	3.9
	DTAEA	**1.4**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
F4	NSGA-II	2.5	2.7	2.7	2.9	2.9	3.1	3.1	3.3
	DNSGA-II	3.2	3.4	3.5	3.7	3.7	3.8	3.9	4.0
	MOEA/D	2.2	2.5	2.6	2.7	2.7	2.8	2.9	2.9
	MOEAD/KF	3.0	3.1	3.1	3.2	3.3	3.5	3.6	3.7
	DTAEA	**2.2**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**

The best rank of MIGD is highlighted in boldface with gray background.

achieve any useful solutions in the 5 generations gap. For F2, DTAEA becomes the best and the rank equals to 1.0 since $\tau_t = 5$. That is simply the opposite situation for F1. F2, adapting from DTLZ2, can easily achieve some, if not all, convergent solutions within 5 generations. Once the DA in DTAEA achieves some solutions away from the CA, DTAEA will become the best. For F3 and F4, DTAEA also performs well. Although the rank of DTAEA is not equal to 1.0 when $\tau_t = 5$, it is always the best. In general, different change frequencies impacts the performance of DTAEA. There will be a maximum change frequency (minimum τ_t) that make DTAEA works. The frequency depends on the test problems. The easier a non-dominate solution is found, the smaller frequency τ_t is. Here it should be pointed out that even the change happens faster than DTAEA can handle, DTAEA is still no worse than other algorithms, as none of the other algorithms can achieve anything in that short period. Larger change frequency has no impact on DTAEA. This is because once DTAEA can achieve some non-dominated solutions away from the original ones, DTAEA works well. Also note that even if τ_t is set to 640, which is relatively large, DTAEA is still the best. The suggestions in [107] only re-evaluate solutions like MOEA/D perform, which is not better than DTAEA. This implies that the suggestions in [107] are not the best choice.

3.4.5 Research Question 5: Effects under Different Change Situation

Noted that in the experiment in Section 3.3, the number of objectives only plus/minus 1 each time. This situation is too simple compared with the real-life scenarios. In Rational Unified Process (RUP) [124] project scheduling problem, the number of objectives is not simply adding or removing one at a time, but adding 3 new objective functions as well as removing 1 existing objective functions. Adding and removing multiple objectives at a time obviously causes difficulty for DTAEA. In this research question, different changing situations are set and the result will be compared.

The first situation is set according to the following $m(t)$:

$$m'(t) = \begin{cases} 3, & t = 1 \\ m(t-1) + 2, & t \in [2, 3] \\ m(t-1) - 2, & t \in [4, 5] \\ 2, & t = 6 \end{cases} \tag{3.15}$$

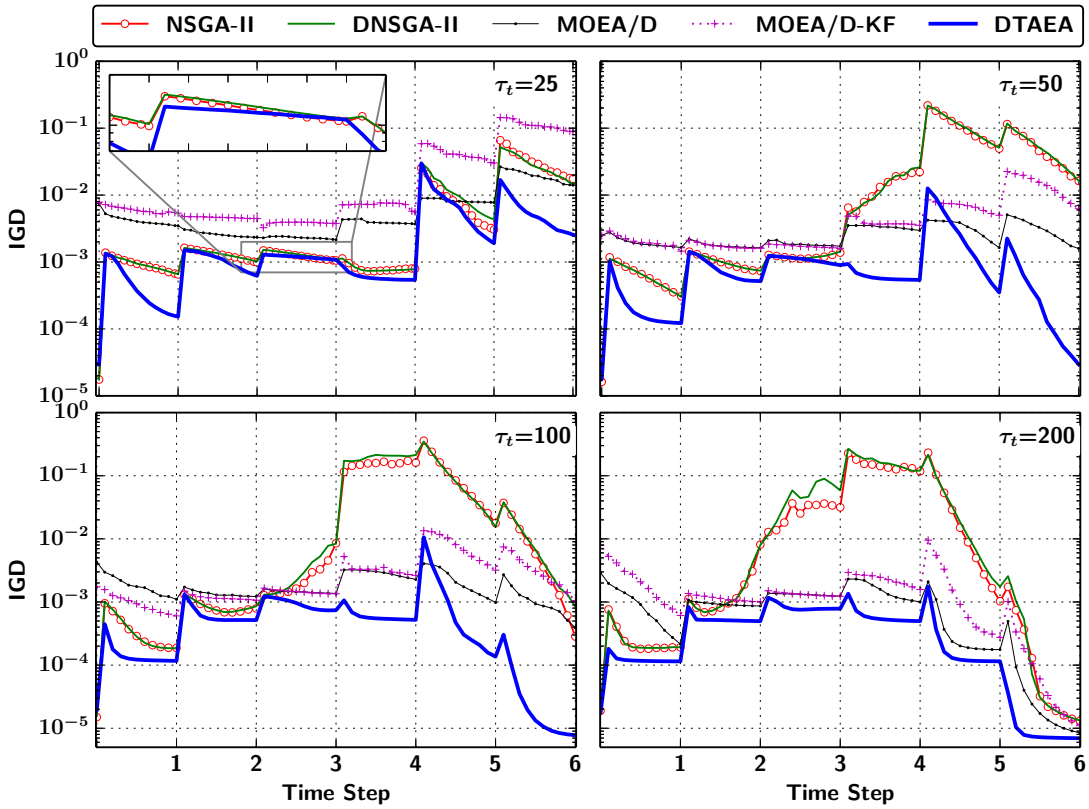
In this situation, the number of objective will increase or decrease 2 objectives at a time, which is more difficult than the setting in Section 3.3.

Table 3.7: Performance Comparisons of DTAEA and its Three Variants on MIGD Metric ($m'(t)$)

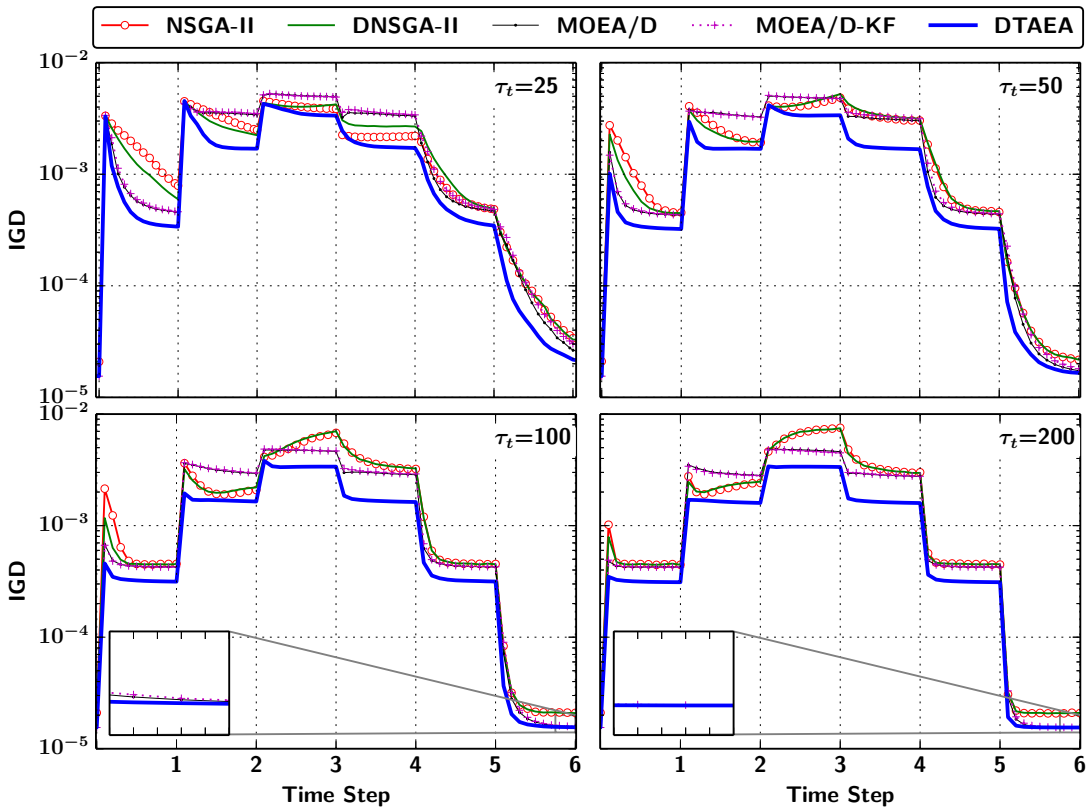
	τ	NSGA-II		DNSGA-II		MOEA/D		MOEA/D-KF		DTAEA	
		MIGD	R	MIGD	R	MIGD	R	MIGD	R	MIGD	R
F1	25	2.97E-3(3.40E-3)†	2.7	2.00E-3(3.56E-3)†	2.8	2.40E-3(8.80E-3)†	3.8	1.67E-2(1.81E-2)†	4.3	**1.10E-3(6.03E-4)**	**1.4**
	50	1.31E-2(1.26E-2)†	3.8	1.33E-2(1.21E-2)†	3.8	7.78E-4(2.92E-4)†	3	1.80E-3(2.10E-3)†	3.5	**3.38E-4(3.38E-5)**	**1**
	100	2.54E-2(3.50E-2)†	4	3.23E-2(3.94E-2)†	4.1	5.73E-4(4.71E-5)†	2.8	7.27E-4(4.18E-4)†	3.2	**2.92E-4(1.75E-6)**	**1**
	200	3.11E-3(4.08E-2)†	4.3	1.25E-2(3.68E-2)†	4.3	5.02E-4(2.16E-4)†	2.4	6.39E-4(1.89E-3)†	3	**2.90E-4(2.10E-6)**	**1**
F2	25	1.41E-3(5.72E-5)†	3.2	1.48E-3(1.11E-4)†	3.4	1.81E-3(8.28E-5)†	3.5	1.82E-3(4.91E-5)†	3.9	**1.07E-3(5.22E-6)**	**1**
	50	1.54E-3(1.01E-4)†	3.8	1.63E-3(1.28E-4)†	4	1.70E-3(6.46E-5)†	2.9	1.71E-3(6.81E-5)†	3.3	**1.06E-3(3.42E-6)**	**1**
	100	1.82E-3(1.32E-4)†	4.1	1.90E-3(9.04E-5)†	4.2	1.62E-3(2.88E-5)†	2.7	1.64E-3(2.74E-5)†	3.1	**1.04E-3(1.83E-6)**	**1**
	200	1.98E-3(1.03E-4)†	4.1	1.96E-3(1.38E-4)†	4.2	1.58E-3(3.77E-5)†	2.8	1.56E-3(2.12E-5)†	2.8	**1.03E-3(1.34E-6)**	**1.1**
F3	25	8.26E-3(9.99E-3)†	2.8	8.09E-3(5.22E-3)†	2.8	4.88E-3(4.31E-3)†	3.8	2.11E-2(4.73E-2)†	4.2	**2.66E-3(3.29E-3)**	**1.4**
	50	1.33E-2(1.64E-2)†	3.4	1.09E-2(1.84E-2)†	3.4	2.79E-3(6.58E-4)†	3.2	5.05E-3(6.62E-3)†	3.8	**1.20E-3(1.04E-4)**	**1.1**
	100	9.02E-2(5.18E-2)†	3.9	9.16E-2(4.43E-2)†	4	2.42E-3(2.19E-3)†	2.8	2.41E-3(1.63E-3)†	3.2	**1.06E-3(8.73E-6)**	**1**
	200	1.51E-1(1.28E-1)†	4.4	1.76E-1(9.13E-2)†	4.4	1.69E-3(5.03E-5)†	2.4	2.01E-3(2.33E-4)†	2.9	**1.05E-3(7.16E-6)**	**1**
F4	25	2.64E-3(4.22E-4)†	2.5	2.79E-3(3.27E-4)†	3	4.08E-3(7.27E-4)†	4.2	3.91E-3(3.17E-4)†	4.3	**1.10E-3(2.29E-5)**	**1**
	50	1.77E-3(3.40E-4)†	2.8	2.26E-3(3.41E-4)†	3.5	3.44E-3(5.08E-4)†	4	3.03E-3(3.74E-4)†	3.8	**1.06E-3(1.24E-3)**	**1**
	100	2.19E-3(2.30E-4)†	3.2	2.29E-3(3.31E-4)†	3.4	3.04E-3(3.42E-4)†	3.8	3.14E-3(3.88E-4)†	3.7	**1.04E-3(2.78E-6)**	**1**
	200	2.26E-3(1.92E-4)†	3.3	2.24E-3(1.88E-4)†	3.1	2.71E-3(1.56E-4)†	3.8	2.68E-3(8.39E-4)†	3.8	**1.03E-3(2.42E-6)**	**1**
F5	25	7.14E-3(3.50E-3)†	3.9	3.87E-3(5.66E-4)†	3.5	2.53E-3(2.74E-4)‡	2.5	**2.30E-3(2.67E-4)‡**	**2.5**	2.94E-3(3.79E-4)	2.7
	50	2.83E-3(2.91E-4)†	3.6	3.27E-3(3.15E-4)†	4.4	1.99E-3(9.74E-5)†	2.6	1.94E-3(1.17E-4)†	2.4	**1.84E-3(8.27E-5)**	**2**
	100	1.72E-3(2.23E-4)†	3	1.84E-3(1.50E-4)†	3.8	1.80E-3(7.46E-5)†	3.3	1.74E-3(1.32E-4)†	3.3	**1.29E-3(1.26E-5)**	**1.6**
	200	1.57E-3(1.12E-4)†	2.8	1.83E-3(1.01E-4)†	4.1	1.68E-3(3.56E-5)†	3.2	1.64E-3(7.30E-5)†	3	**1.19E-3(1.29E-5)**	**1.9**
F6	25	8.61E-3(1.74E-3)†	4.4	5.03E-3(5.70E-4)†	3.5	2.58E-3(3.54E-4)‡	2.3	**2.57E-3(3.77E-4)‡**	2.6	3.04E-3(1.71E-4)	**2.2**
	50	2.92E-3(2.43E-4)†	4	3.72E-3(6.24E-4)†	3.8	2.24E-3(2.14E-4)†	2.5	2.38E-3(1.57E-4)†	2.7	**2.05E-3(1.03E-4)**	**1.9**
	100	2.69E-3(3.24E-4)†	3.9	2.62E-3(1.58E-4)†	4.3	1.89E-3(9.26E-5)†	2.5	1.98E-3(1.20E-4)†	3	**1.35E-3(1.34E-5)**	**1.4**
	200	2.89E-3(3.07E-4)†	4.3	2.53E-3(1.01E-4)†	4	1.87E-3(7.80E-5)†	2.4	1.85E-3(7.47E-5)†	2.9	**1.27E-3(2.42E-5)**	**1.4**

R denotes the global rank assigned to each algorithm by averaging the ranks obtained at all time steps. Wilcoxon's rank sum test at a 0.05 significance level is performed between DTAEA and each of NSGA-II, DNSGA-II, MOEA/D and MOEA/D-KF. † and ‡ denote the performance of the corresponding algorithm is significantly worse than and better than that of DTAEA, respectively. The best median value is highlighted in boldface with grey background.

As shown in Table 3.7 and Table 3.8, the performance of DTAEA is still robust as it obtains the best MIGD and MHV values under all four frequencies of change. Figure 3.15(a) shows the trajectories of IGD values obtained by different algorithms across the whole evolution process; while Figure 3.18(a) shows the average ranks of IGD obtained by different algorithms at each time step. From these two figures, it can be clearly seen

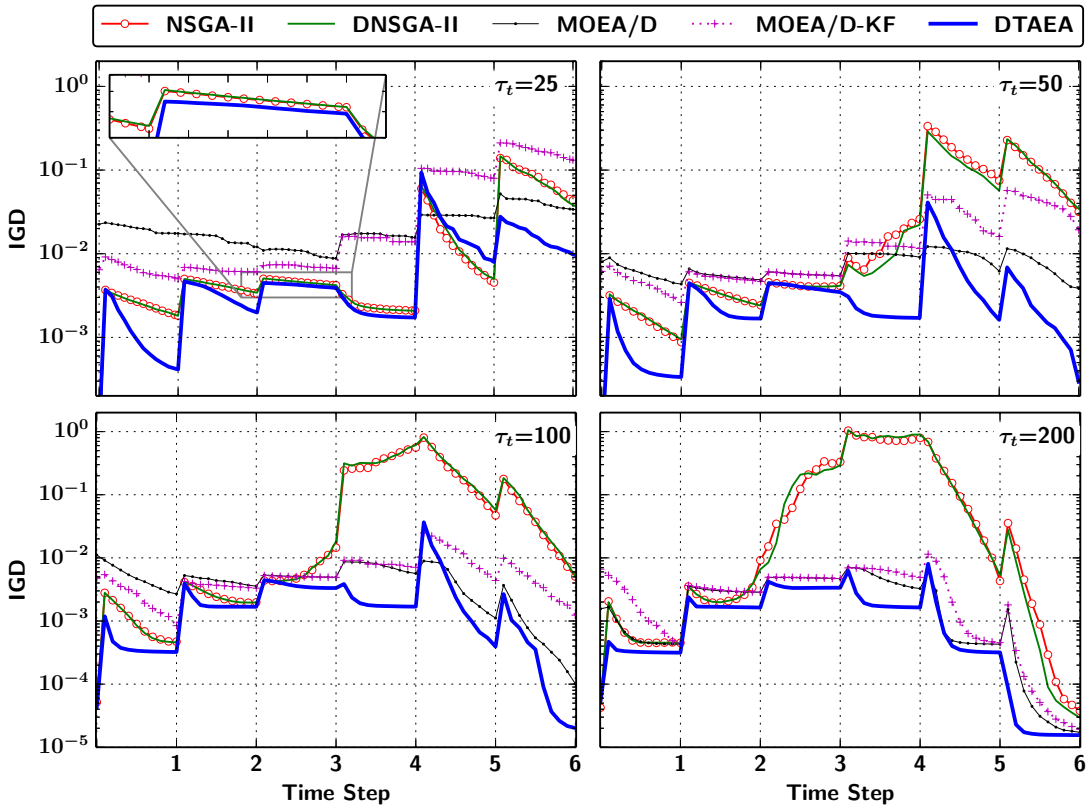


(a) F1

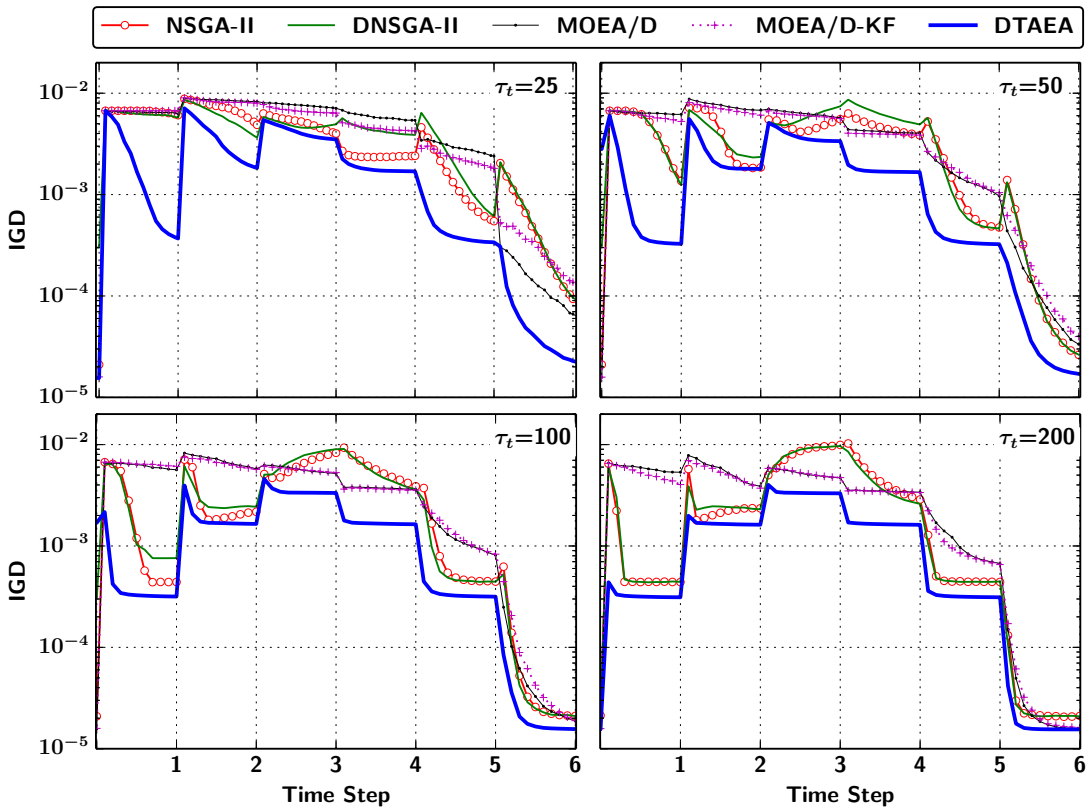


(b) F2

Figure 3.15: IGD trajectories across the whole evolution process($m'(t)$)(F1,F2).

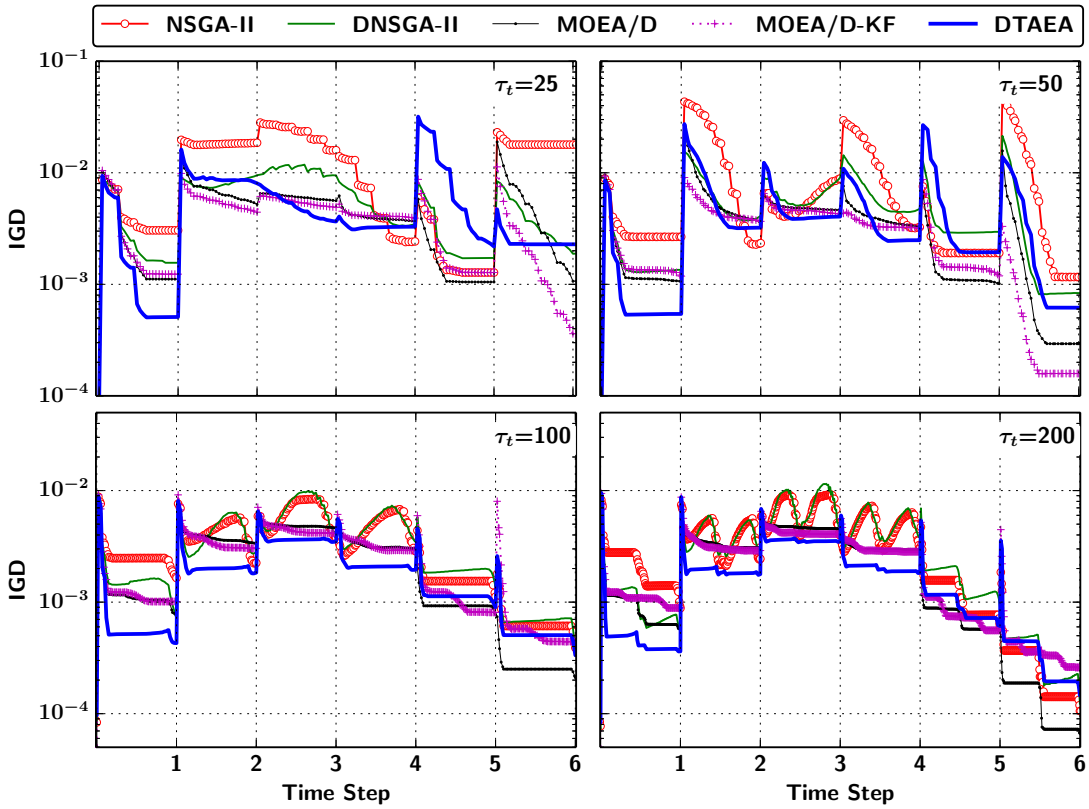


(a) F3

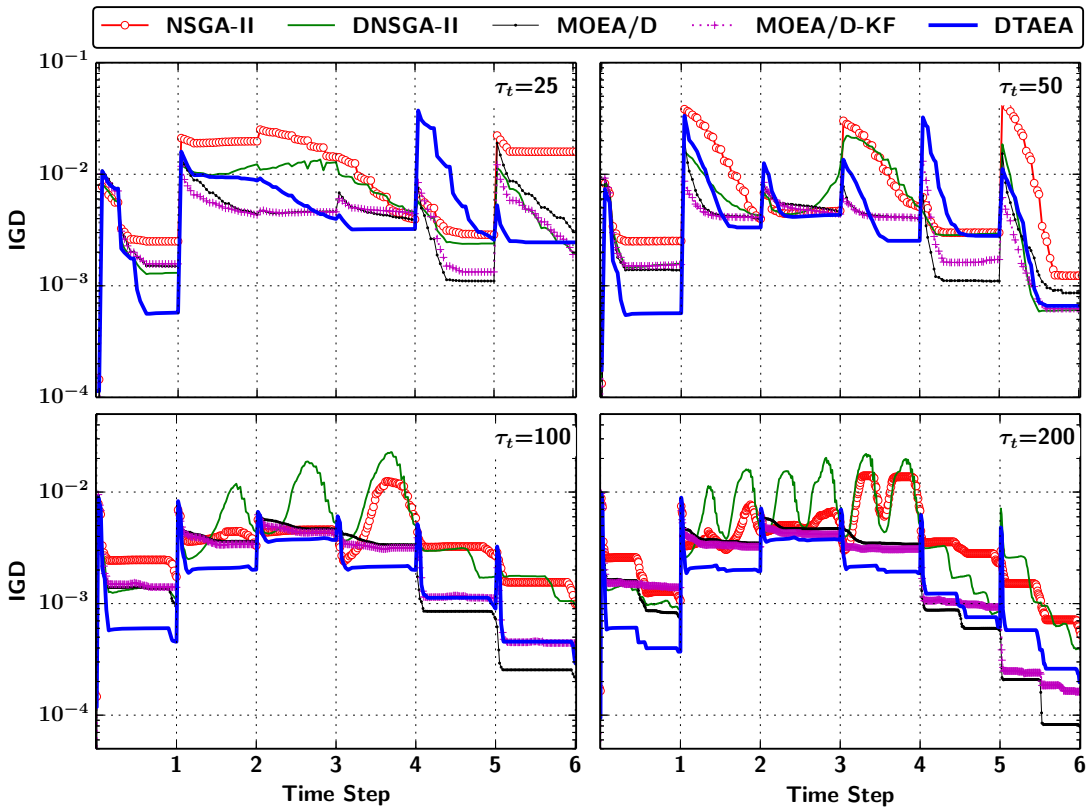


(b) F4

Figure 3.16: IGD trajectories across the whole evolution process($m'(t)$)(F3,F4).

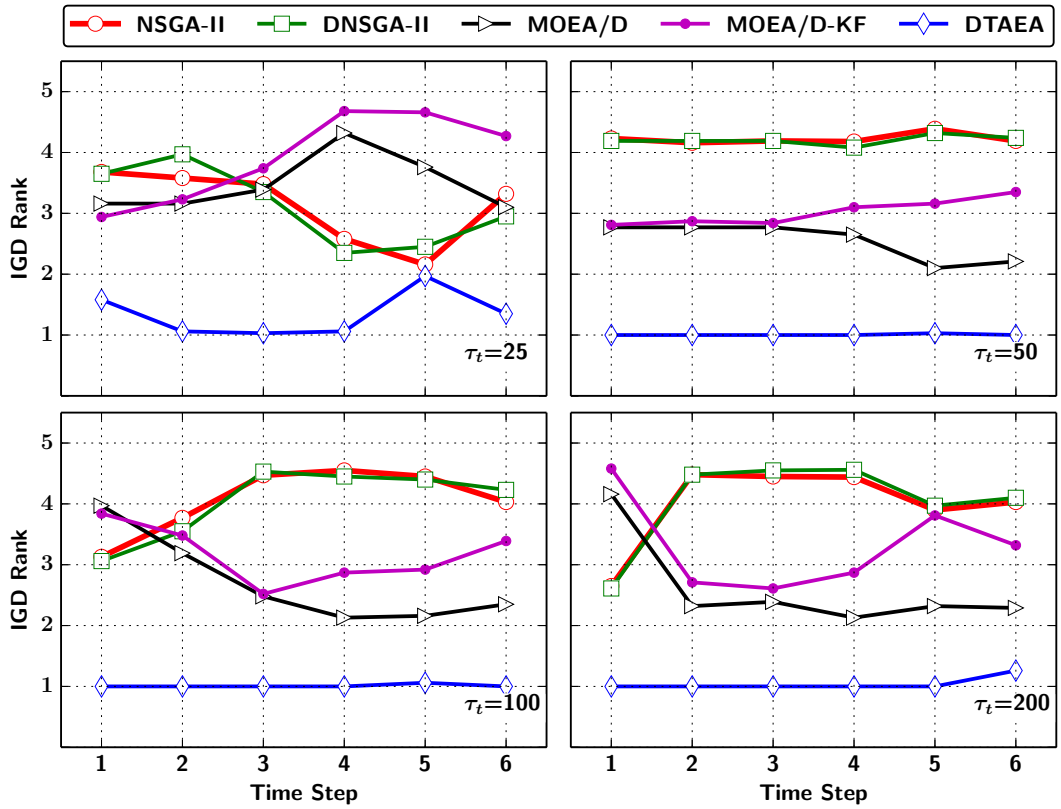


(a) F5

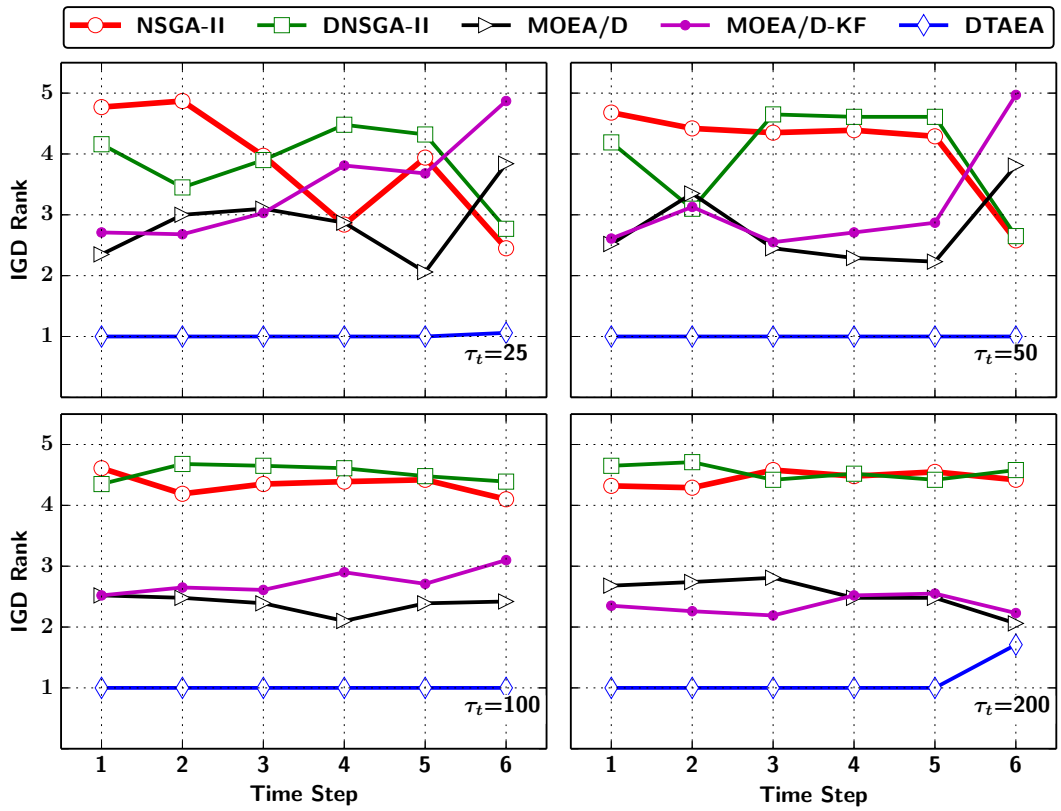


(b) F6

Figure 3.17: IGD trajectories across the whole evolution process($m'(t)$)(F5,F6).

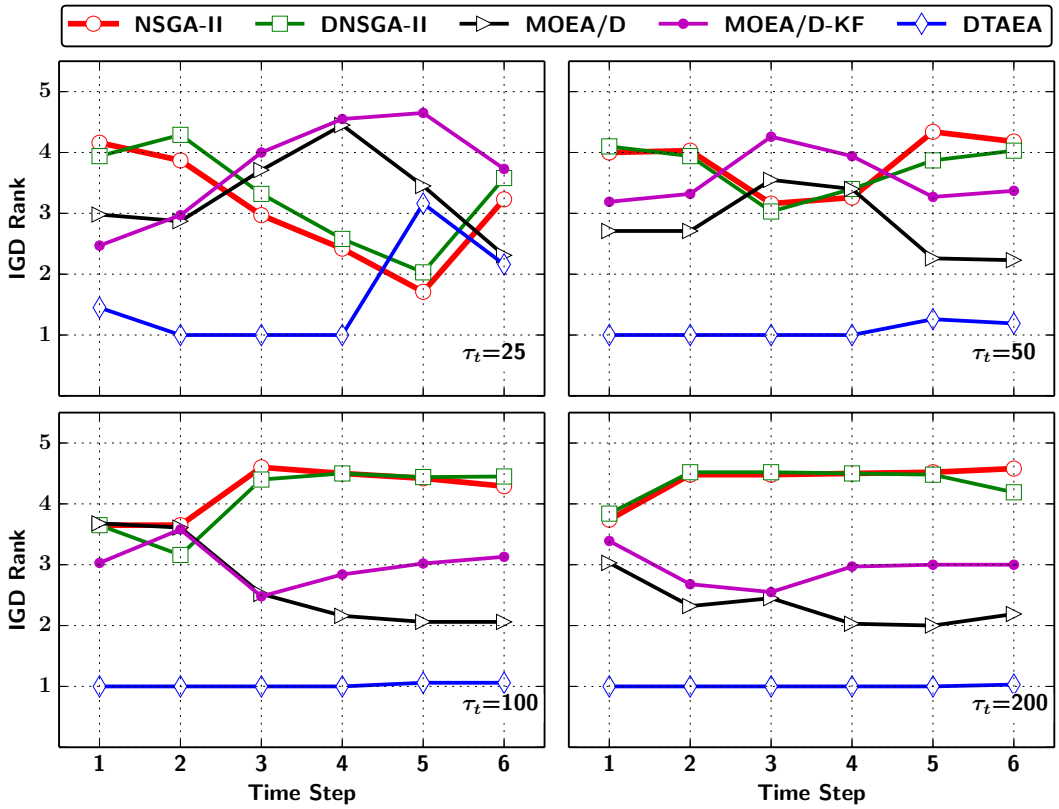


(a) F1

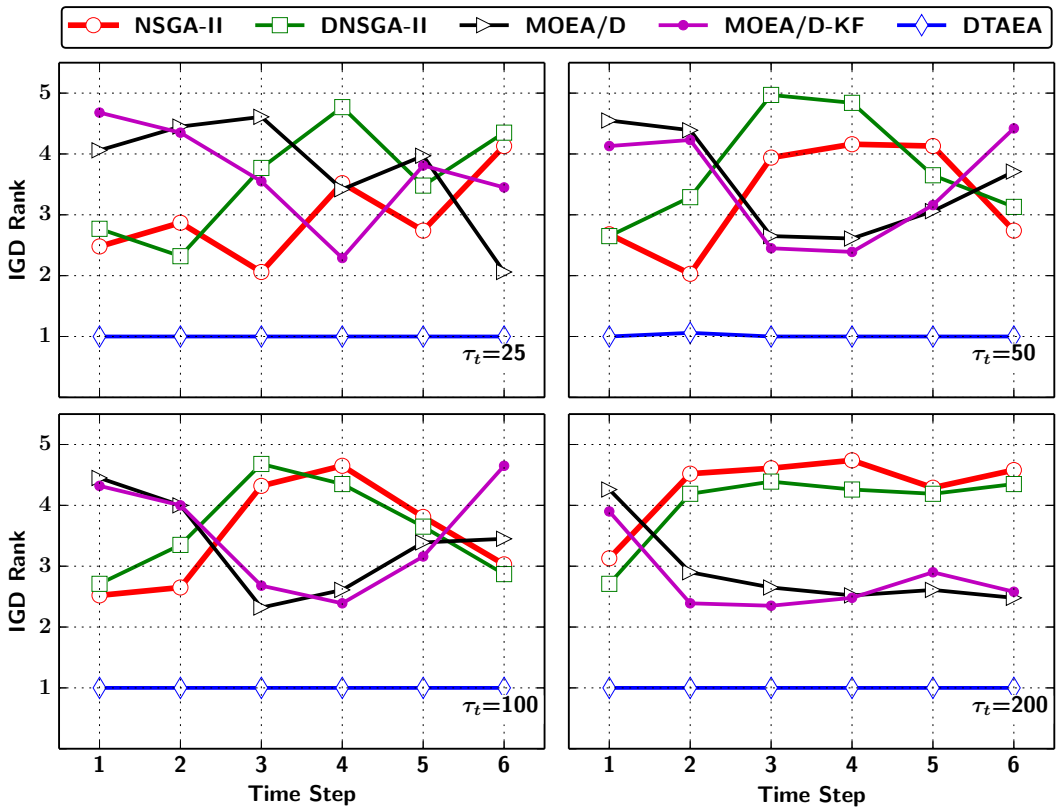


(b) F2

Figure 3.18: The rank of IGD obtained by different algorithms at each time step($m'(t)$) (F1,F2).

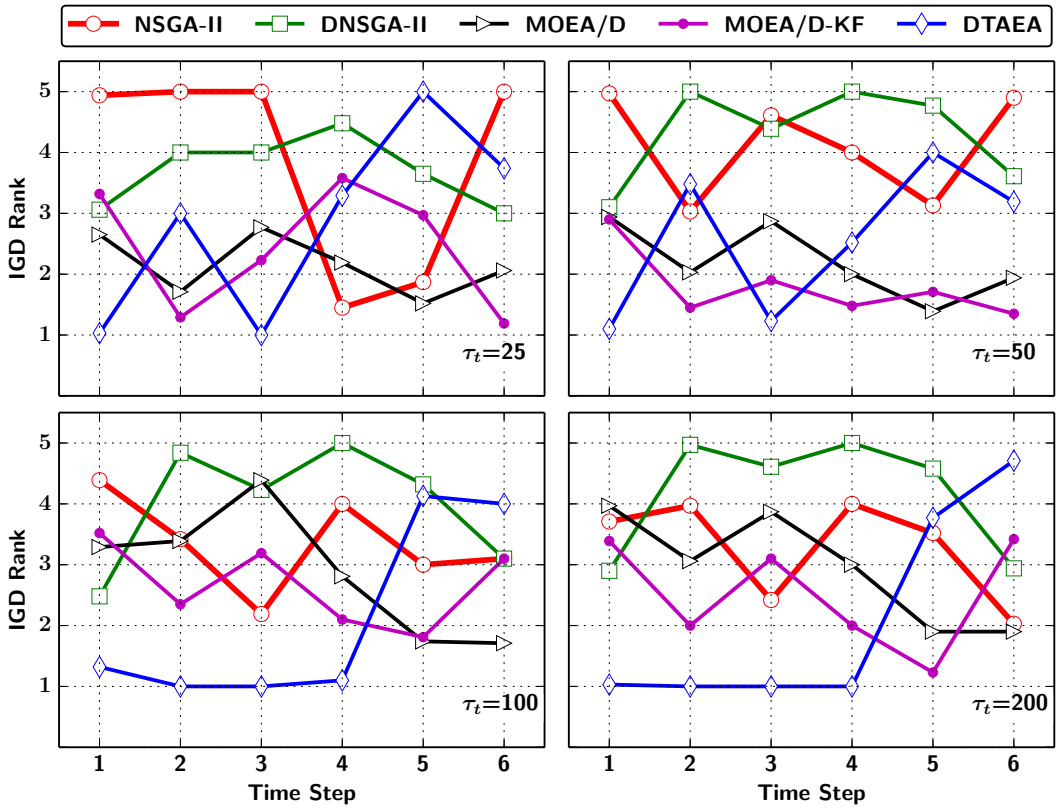


(a) F3

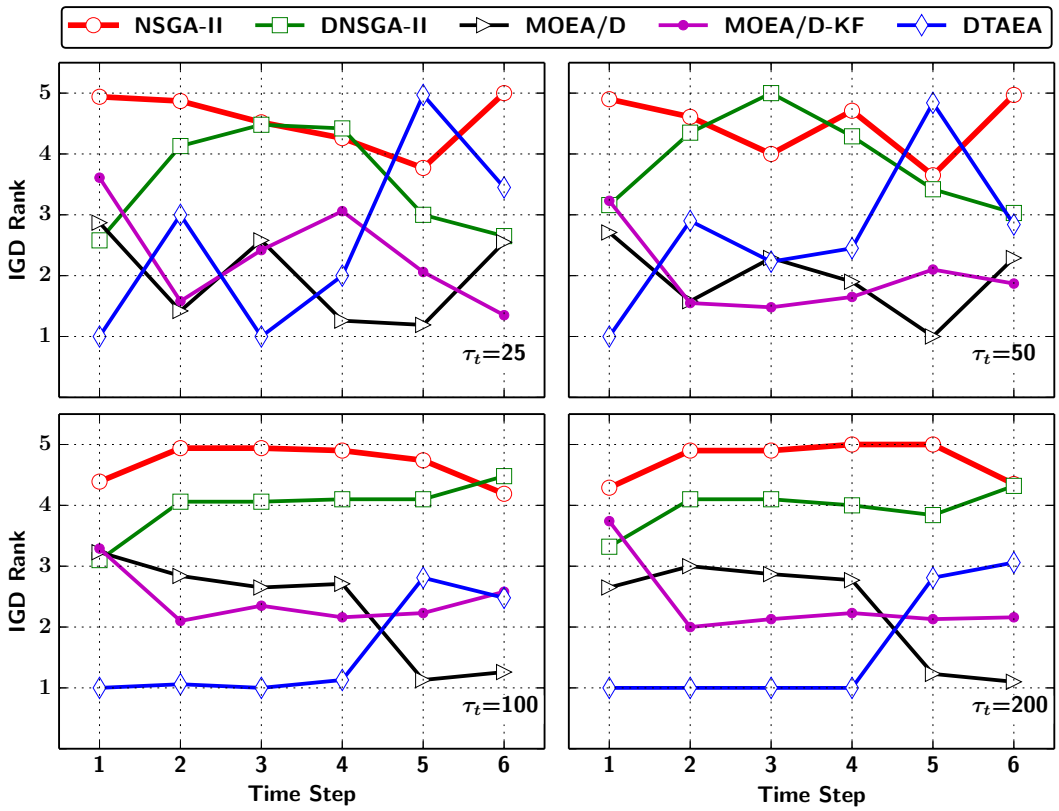


(b) F4

Figure 3.19: The rank of IGD obtained by different algorithms at each time step($m'(t)$)(F3,F4).



(a) F5



(b) F6

Figure 3.20: The rank of IGD obtained by different algorithms at each time step($m'(t)$)(F5,F6).

Table 3.8: Performance Comparisons of DTAEA and its Three Variants on MHV Metric($m'(t)$)

	τ	NSGA-II		DNSGA-II		MOEA/D		MOEA/D-KF		DTAEA	
		MHV	R	MHV	R	MHV	R	MHV	R	MHV	R
F1	25	87.4%(1.13E-1)†	3.1	92.5%(1.33E-1)†	3.1	91.6%(3.74E-1)†	3.5	70.2%(1.77E-1)†	3.9	**98.2%(2.31E-2)**	**1.3**
	50	66.4%(1.49E-1)†	4.2	63.6%(2.28E-1)†	4.2	99.4%(7.77E-3)†	2.5	92.7%(1.24E-1)†	3	**100.0%(1.26E-4)**	**1**
	100	71.8%(1.12E-1)†	4.1	71.4%(8.87E-2)†	4	99.9%(1.26E-3)†	2.7	99.5%(1.13E-2)†	3.2	**100.0%(5.61E-5)**	**1**
	200	84.8%(1.97E-1)†	4	82.1%(2.41E-1)†	4	100.0%(1.08E-2)†	2.6	99.7%(9.86E-2)†	3.3	**100.0%(1.73E-5)**	**1**
F2	25	91.1%(3.29E-3)†	3.8	91.5%(2.75E-3)†	3.8	92.1%(1.93E-3)†	2.9	92.0%(1.49E-3)†	3.5	**92.9%(1.06E-4)**	**1**
	50	91.6%(3.16E-3)†	4.1	91.6%(3.23E-3)†	4	92.3%(1.01E-3)†	2.8	92.3%(1.28E-3)†	3.1	**92.9%(5.67E-5)**	**1**
	100	90.6%(8.76E-3)†	4.3	90.3%(8.01E-3)†	4.5	92.5%(6.31E-4)†	2.4	92.4%(6.47E-4)†	2.7	**92.9%(1.65E-5)**	**1**
	200	89.7%(7.17E-3)†	4.4	89.8%(9.25E-3)†	4.5	92.5%(5.50E-4)†	2.5	92.6%(4.09E-4)†	2.3	**93.0%(5.51E-6)**	**1.1**
F3	25	76.0%(5.96E-2)	3.1	75.8%(1.80E-2)	3.3	78.4%(3.14E-1)†	3.3	62.2%(6.96E-2)†	3.7	**85.5%(1.92E-1)**	**1.6**
	50	63.2%(1.23E-1)†	3.8	62.9%(2.14E-1)†	3.7	86.8%(3.79E-2)†	2.8	74.3%(2.87E-1)†	3.6	**92.5%(3.81E-3)**	**1.1**
	100	52.5%(9.26E-2)†	4.2	54.2%(5.61E-2)†	4.1	87.6%(1.24E-1)†	2.7	88.0%(1.04E-1)†	3	**92.9%(4.27E-4)**	**1**
	200	60.7%(6.73E-2)†	4.4	60.3%(7.66E-2)†	4.3	92.2%(3.02E-3)†	2.3	90.7%(2.05E-2)†	2.9	**92.9%(3.01E-4)**	**1**
F4	25	90.3%(5.61E-3)†	3	89.5%(5.88E-3)†	3.6	87.5%(4.24E-2)†	3.8	88.7%(1.84E-2)†	3.7	**92.9%(1.65E-4)**	**1**
	50	91.3%(1.32E-2)†	3.3	89.2%(1.91E-2)†	3.8	89.9%(2.30E-2)†	3.5	90.6%(5.72E-3)†	3.5	**92.9%(4.48E-2)**	**1**
	100	88.9%(1.26E-2)†	3.5	88.2%(2.45E-2)†	3.6	90.5%(3.51E-3)†	3.4	90.4%(9.23E-3)†	3.5	**93.0%(9.81E-6)**	**1**
	200	87.2%(1.65E-2)†	4.3	87.7%(1.92E-2)†	4	90.7%(5.10E-4)†	2.9	90.7%(1.84E-2)†	2.8	**93.0%(4.10E-6)**	**1**
F5	25	55.6%(5.86E-2)†	3.9	77.3%(3.98E-2)†	3.7	90.3%(1.23E-2)‡	**2.2**	**91.0%(4.75E-3)**‡	2.4	84.7%(2.59E-2)	2.8
	50	83.9%(1.80E-2)†	4.1	83.3%(2.76E-2)†	4.3	91.8%(3.26E-3)	2.2	**91.9%(4.21E-3)**	**1.8**	91.2%(2.63E-3)	2.6
	100	91.4%(8.99E-3)†	3.3	91.7%(3.53E-3)†	4	92.2%(2.68E-3)†	2.9	92.1%(5.46E-3)†	2.7	**92.3%(4.12E-4)**	**2.1**
	200	92.1%(5.78E-3)†	3.3	91.6%(3.04E-3)†	4.2	92.4%(5.56E-4)†	3	92.3%(3.23E-3)†	2.5	**92.7%(2.17E-4)**	**2.1**
F6	25	54.3%(5.32E-2)†	4.6	68.8%(3.85E-2)†	3.5	90.8%(1.37E-2)‡	**2**	**91.1%(7.15E-3)**‡	2.3	85.2%(1.40E-2)	2.6
	50	86.6%(1.04E-2)†	4.5	80.1%(4.06E-2)†	3.9	**91.8%(6.93E-3)**‡	**2**	91.7%(4.34E-3)‡	2	90.7%(3.66E-3)	2.7
	100	88.4%(1.29E-2)†	4.7	90.5%(3.61E-3)†	4	92.2%(1.36E-3)†	2.3	92.1%(4.34E-3)†	2.5	**92.3%(4.21E-4)**	**1.6**
	200	88.3%(1.27E-2)†	4.7	90.9%(3.60E-3)†	3.9	92.4%(8.28E-4)†	2.3	92.3%(2.97E-3)†	2.4	**92.6%(4.68E-4)**	**1.6**

R denotes the global rank assigned to each algorithm by averaging the ranks obtained at all time steps. Wilcoxon's rank sum test at a 0.05 significance level is performed between DTAEA and each of NSGA-II, DNSGA-II, MOEA/D and MOEA/D-KF. † and ‡ denote the performance of the corresponding algorithm is significantly worse than and better than that of DTAEA, respectively. The best median value is highlighted in boldface with gray background.

that DTAEA shows the best performance at most of the time steps. Unlike previous experiments, the performance of DTAEA has stronger fluctuations when $\tau_t = 25$. This is because, under a high frequency of change, it will be even harder for DTAEA to drive the solutions fully covering the PF_t before the environment changes. As the number of objectives increases/decreases 2 at a time, the solution will be even more crowded, which means pushing solutions away may be more difficult. When the frequency of change decreases, i.e., the increase of τ_t, the performance of DTAEA becomes stable again. Under a high frequency of change, i.e, $\tau_t = 25$, NSGA-II and MOEA/D have still shown a better performance than DNSGA-II and MOEA/D-KF at every time step, which is consistent with the conclusion from the previous experiment. The rest of sub-figures in Figure 3.15, Figure 3.16, Figure 3.18 and Figure 3.19 show that DTAEA is still the best on all time steps on F1For F2,F3 and F4 since $\tau_t = 50$.

For the problem of F5 and F6, the performance of DTAEA sharply decreases. DTAEA only becomes the best again after $\tau_t = 50$. The reason is complicated. In F5 and F6, there are two types of changes occurring. The change in the number of objectives is much more

dramatic than the moving of PS. Although DTAEA does not react to the moving of PS, which actually leads to loss of performance. However, DTAEA gains the advantage back because it is the only algorithm that can handle the changing number of objectives. In the situation with $m'(t)$, the performance DTAEA still beats (but no longer overwhelmingly) other algorithms. The pitfalls of DTAEA not reacting towards the moving of PS becomes dominating. This leads to the sharply decreasing performance of DTAEA. To fix this problem, DTAEA needs to react to the moving of PS. Which can be studied in the future works.

Generally speaking, the effects caused by the $m'(t)$ brings challenges to all the algorithms. As DTAEA relies on the information of the archives before changes, the larger the changes are, the less useful those archives are. That is because when increasing multiple objectives at a time, the solutions will be squeezed into even smaller subspace. The mating strategy of DTAEA always selects one parent from the CA, which brings a difficulty in pushing solutions away from their current locations.

3.4.6 Research Question 6: Effects under Non-partitioned Variables

The test problems for dynamic multi-objective optimization with a changing number of objectives are extended from scalable test problems for MOPs. However, most of the existing scalable test problems for MOPs are based on a generic multi-objective test problem generator proposed by Deb et al. [84]. The variables of the problems generated by this generator are partitioned. To be more specific, for a problem with m objective functions, the decision variable vector is partitioned into M non-overlapping groups:

$$\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{\mathbf{m}-1}, \mathbf{x}_{\mathbf{m}})^T \tag{3.16}$$

The test problem is suggested as:

$$\left.\begin{array}{rl} \text{Minimize} & f_1(\mathbf{x}_1), \\ \text{Minimize} & f_2(\mathbf{x}_2), \\ \vdots & \vdots \\ \text{Minimize} & f_{m-1}(\mathbf{x_{m-1}}), \\ \text{Minimize} & g(\mathbf{x_m})h(f_1(\mathbf{x}_1), f_2(\mathbf{x}_2), ..., f_{m-1}(\mathbf{x_{m-1}})), \\ \text{subject to} & \mathbf{x} \in \Omega \end{array}\right\} \tag{3.17}$$

In this case, we consider that this problem has partitioned variables. If the variables of a problem cannot be partitioned into several non-overlapping groups, we consider that this problem has non-partitioned variables. In this research question, we focus on the performance of DTAEA handling problems have non-partitioned variables.

According to the conclusion from Deb [84], any solutions on the Pareto-optimal front (PF) for problems generated by this generator must be with $g(\mathbf{x_m})$ achieving the global optimal g^*. The proof is:

Proof. Assumed that $\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x_{m-1}}, \mathbf{x_m}\}$ do not achieve global optimal g^*, there exists $\hat{\mathbf{x}} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x_{m-1}}, \hat{\mathbf{x}}_\mathbf{m}\}$, which $g(\hat{\mathbf{x}}_\mathbf{m}) = g^*$. According to the assumption, $g(\hat{\mathbf{x}}_\mathbf{m}) = g^* < g(\mathbf{x_m})$. So we have $f_1(\mathbf{x}) = f_1(\hat{\mathbf{x}}), f_2(\mathbf{x}) = f_2(\hat{\mathbf{x}}), ..., f_{m-1}(\mathbf{x}) = f_{m-1}(\hat{\mathbf{x}})$, while $f_m(\hat{\mathbf{x}}) = g(\hat{\mathbf{x}})h(\mathbf{x}) < g(\mathbf{x})h(\mathbf{x}) = f_m(\mathbf{x})$. $\mathbf{f}(\hat{\mathbf{x}}) \prec \mathbf{f}(\mathbf{x})$, $\mathbf{f}(\mathbf{x})$ is not on the Pareto-optimal front. ■

From this conclusion, we can divide the decision variables into two groups: variables $\mathbf{x_m}$ decides whether the solution is on the PF, and variables $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x_{m-1}}\}$, denoted as diversity variables, only impacts on the position of the solution located on the PF, if the solution is on the PF. According to the conclusion from Deb and etc. [84], the number of the diversity variables is $m-1$ and the rest $n-m+1$ variables are convergence variables. Partitioned variables have a great impact on the analysis of solution location after increasing or decreasing number of solutions. The reason is that the $n-m(t-1)+1$

convergence variables $\mathbf{x_{t-1}}$ of Pareto-optima solution before changes must converge to a value to achieve the global optimal g^*. When the number of objective increases ($m(t) > m(t-1)$), the number of the convergence variables decreases. This solution is still Pareto-optima after the changes. In the meantime, those diversity variables, which originally were convergence variables, are converged in a value, which leads to a bad diversity after the changes.

Obviously, the assumption of partitioned variables is not what always is happening in real-life scenarios. In real-life scenarios, every variable is possible impacting the location on the PF as well as the distance from the solution to the PF. To further understand the characteristic of DMOPs with a changing number of objectives, in this research question, we will discuss the situation when dealing with problems with non-partitioned variables. We try to create problems with non-partitioned variables by "mix up" partitioned variables. The following equations give three mixing functions generating problems with non-partitioned variables.

Function $\mathbf{g}_1(\mathbf{x})$ linearly mixes up variables:

$$\begin{aligned} \text{minimize} \quad & \mathbf{f}(g_1(\mathbf{x})) \\ \text{with} \quad & g_1(\mathbf{x}) = \{\tfrac{x_1+x_m}{2}, \tfrac{x_2+x_{m+1}}{2}..., \tfrac{x_n+x_{(m+n)\%n+1}}{2}\} \end{aligned} \tag{3.18}$$

In which m is the number of objectives and n is the number of variables. As $x_1, x_2, ..., x_{m-1}$ are diversity variables while $x_m, x_{m+1}, ..., x_n$ are convergence variables, the $x'_1, ..., x'_{m-n}$ are either diversity variables or convergence variables.

Function $\mathbf{g}_2(\mathbf{x})$ and Function $\mathbf{g}_3(\mathbf{x})$ non-linearly mixes up variables:

$$\begin{aligned} \text{minimize} \quad & \mathbf{f}(g_2(\mathbf{x})) \\ \text{with} \quad & g_2(\mathbf{x}) = \{x_1 x_{\frac{n+1}{2}}, x_2 x_{\frac{n+2}{2}}, ..., x_n x_{\frac{n+n}{2}}\} \end{aligned} \tag{3.19}$$

$$\begin{aligned} \text{minimize} \quad & \mathbf{f}(g_3(\mathbf{x}')) \\ \text{with} \quad & g_3(\mathbf{x}) = \{sin(x_1) x_{\frac{n+1}{2}}, ..., sin(x_n) x_{\frac{n+n}{2}}\} \end{aligned} \tag{3.20}$$

in which $\mathbf{x} = \{x_1, x_2, ..., x_n\}$ are the original partitioned variables. After mixing the variables, the variables cannot be divided into two groups any more.

We use the same analysing method of Section 3.1 to analyse the location of the solutions after the changes. A good approximation to the Pareto-optimal front of two-objective DTLZ2 is obtained from a 200 generations' run with the MOEA/D [111] at first. Then the underlying problem changes to a three-objective instance. Figure 3.21 shows the distribution of the population gained after re-evaluating the objective values. From the Figure 3.21 we can clearly find the conclusion mention above is correct. The solutions after the changes are still located on the PF after increasing the number of objectives but the diversity of the solutions is bad.

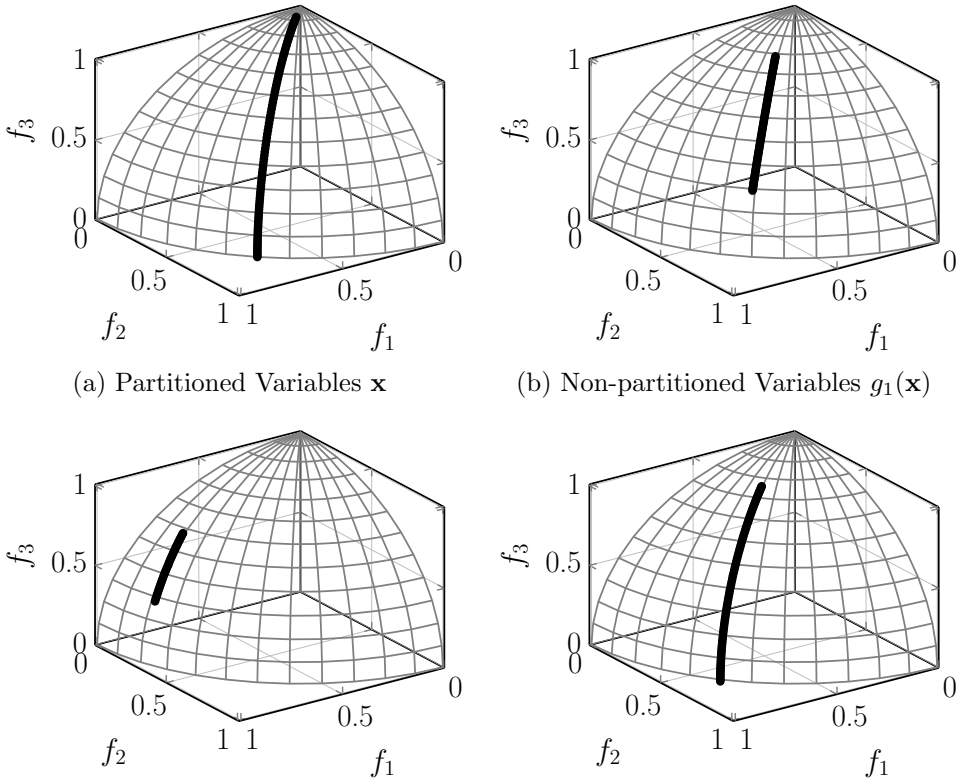
The explanation for this phenomenon is as followed. For an m-objectives optimization problem, if all the objectives conflict with each other, the PF should be on a $m-1$ hypesurface according to Deb's work [84]. Locating points on the new curve requires $m-1$ independent variables, named $\mathbf{x}' = \{x'_1, x'_2..., x'_{m-1}\}$. There always exists a mapping $\mathbf{F} : \Omega \to \Omega$, which maps $\mathbf{x}$ to $\mathbf{x}' = \{x'_1, x'_2, ...x'_{m-1}, ...x'_n\}$. The mapping $\mathbf{F}$ impacts the location and the shape of the curve. However, $n - m + 1$ independent variables named $\{x_m, x_{m+1}, ..., x_n\}$ still play the same role as convergence variables. In conclusion, the problems with non-partitioned variables have the same feature of those with partitioned variables.

The experimental results about the comparison of five algorithms is in the Table 3.9. The $\tau = 100$ for every runs. The result shows that DTAEA is always the best even dealing with non-partitioned variables.

Table 3.9: Performance Comparisons of DTAEA dealing problem with 3 types of non-partitioned variables on IGD Metric($m'(t)$)

	NSGA-II		DNSGA-II		MOEA/D		MOEA/D-KF		DTAEA	
	MIGD	R	MIGD	R	MIGD	R	MIGD	R	MIGD	R
g_1	1.92E-3(1.27E-4)†	3.2	2.02E-3(2.12E-4)†	3.6	2.19E-3(1.17E-4)†	3.3	2.26E-3(6.71E-5)†	3.8	**1.26E-3(4.58E-6)**	**1.1**
g_2	2.20E-3(9.07E-5)†	3.6	2.24E-3(1.47E-4)†	3.7	2.08E-3(5.81E-5)†	3.2	2.10E-3(6.03E-5)†	3.5	**1.25E-3(3.26E-6)**	**1**
g_3	2.45E-3(1.37E-4)†	3.8	2.47E-3(1.41E-4)†	3.9	2.01E-3(2.89E-5)†	3.1	1.99E-3(3.51E-5)†	3.1	**1.22E-3(1.30E-6)**	**1**

R denotes the global rank assigned to each algorithm by averaging the ranks obtained at all time steps. Wilcoxon's rank sum test at a 0.05 significance level is performed between DTAEA and each of NSGA-II, DNSGA-II, MOEA/D and MOEA/D-KF. † and ‡ denote the performance of the corresponding algorithm is significantly worse than and better than that of DTAEA, respectively. The best median value is highlighted in boldface with gray background.



(a) Partitioned Variables $\mathbf{x}$ (b) Non-partitioned Variables $g_1(\mathbf{x})$

(c) Non-partitioned Variables $g_2(\mathbf{x})$ (d) Non-partitioned Variables $g_3(\mathbf{x})$

Figure 3.21: Comparison of population distributions when decreasing the number of objectives with mixed variables.

3.4.7 Research Question 7: Many-Objective Optimization Based on a Changing Number of Objective Approach

Many-objective optimisation brings great challenges to evolutionary algorithms. Usually, optimisation problems with more than three objectives are denoted as many-objective optimisation problems. In real-life scenarios, many-objective optimisation problems occur widely in many cases, such as water network distributing problem [125] and software project scheduling problems. A wide variety of evolutionary algorithms have been proposed to handle many-objective optimisation problems. Some works put efforts on the improvement of the existing algorithms for MOPs, while others develop new techniques for many-objective optimisation problems. In this section, we propose a new method for many-objective optimization: solving many-objective optimization problems in a changing number of objective way. The basic idea of the method is described as follows.

When solving an n-objective optimization problem, two objectives of these n objectives are selected to form a two-objective optimization problem at the beginning. The algorithm attempts to find a well-converged and well-distributed population for this two-objective problem. Once the algorithm achieves a satisfying population, one objective will be added. This process repeats until the all the objectives are added. The flowchart can be found in Figure 3.22.

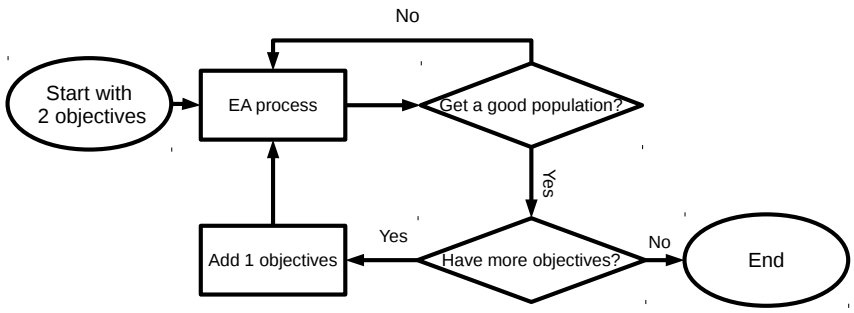


Figure 3.22: Flow chart of solving many-objectives in a changing number of objectives way.

The advantage of using DTAEA to solve many-objective is clear. With the help of the lower dimension solutions, DTAEA can explore the search spacequicker than other algorithms. Finding a well-converged and well-distributed population in two dimensions

is easier comparing to the situation in the high dimension. The well-converged solutions achieved in the lower dimension can guide the searching process in the higher dimension.

However, the challenge is huge. The main challenge is discussed as follows.

- Except for the indicator-based algorithms, most algorithms have no run-time performance indicator which indicates the situation of the population during the optimization process. A performance indicator is needed, otherwise in no way algorithms can know whether the population is well-converged and well-distributed. As a result, the algorithms cannot decide when to add a objective. As discussed in Chapter 2, the performance indicators are normally costly, especially when dealing problems with many objectives. For example, it costs days for SMS-EMOA to solve a 15-objective DTLZ2. How to measure the "quality" of the population in the high dimension problem is very difficult. To overcome this challenge, a well-designed population estimation method should be designed. Also, Current algorithms usually stop when the number of function evaluations exceed a certain amount. However, based on Figure 3.22, the number of generation needed depends on the quality of the population. By adjusting the estimation method mentioned above, it is possible to adjust the total number of evaluations. We should point out that the tuning of the parameters in the estimation method will become very challenging if the number of function evaluations is set.

Considering the challenges mentioned above, we adapt DTAEA to solve many-objective optimization problems.

One simple method dealing with the challenges above is by using pre-setting parameters. For example, let the DTAEA run 400 generations before adding the first objective, and then add one objective every 100 generation until all 15 objectives are added. Because evaluating a two-objective problem costs obviously less than evaluating a 15-objective problem, we count the number of function evaluations in a different way: evaluating a solution for n-objective problems costs n function evaluations. The total number of function evaluations for this process mentioned above is 337,900 (assumed the population size

is 100). The Figure 3.23 shows the IGD result for DTLZ2. From the figure, it can be found that the DTAEA seems to find a well-distributed and well-converged population for a 15-objective problem. We can find out that DTAEA can be use for solving many-objective problems. However, the main drawback of the pre-setting method is that for every problem, we need to adjust the pre-setting parameters, which is difficult.

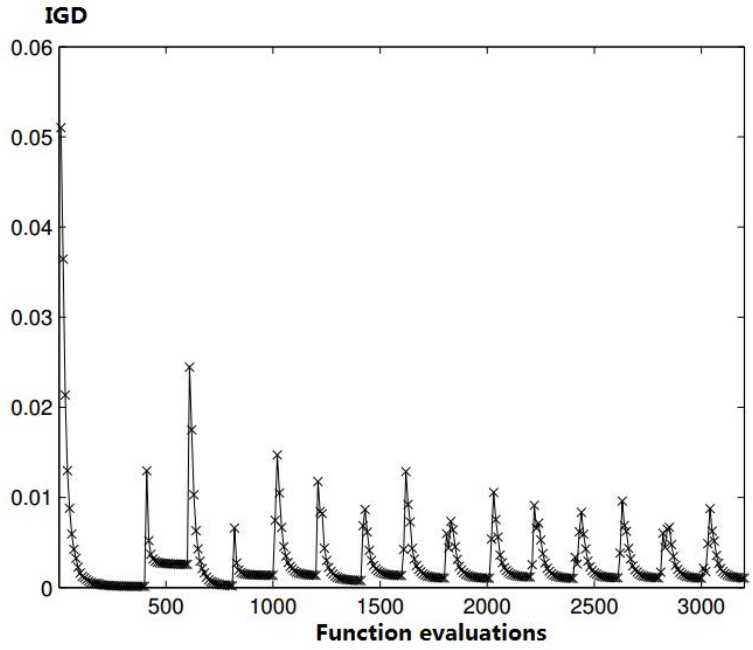


Figure 3.23: IGD for DTLZ2(two-obj to 15-obj)

Another adaptive method is using the improvement for the geometric centre of the population. The pseudo-code is shown as the Algorithm 8. We use the the geometric centre of DA as an example. After updating DA, the current geometric centre point p_t of DA is calculated. Then this point is compared to the one in the previous generation. If $|p_t - p_{t-1}| < \epsilon$, one objective will be added. DTLZ series is selected as the test problem and the number of objectives is set as 15. All the setting for DTAEA and MOEA/D is the same in line with Section 3.3. The following Table 3.10 shows the results for DTAEA and MOEA/D. The ϵ is set as 0.5,0.1,0.01 and 0.001.

According to the table, DTAEA can handle many-objective optimization problem. Moreover, DTAEA wins 11 out of 14 cases compared to MOEA/D.

Algorithm 8: Mating Selection Mechanism

Input: initial population $\mathbf{P_{ini}}$,threshold ϵ,number of objective O
Output: population $\mathbf{P}$

```
P ← P_ini;
q =QualityEstimation(P);
O ← 2;
repeat
    repeat
        P ← DTAEA(P);
        q =QualityEstimation(P);
    until q < ε;
    Add one objective to the test problem;
    O ← O + 1;
until O < M;
return P
```

Table 3.10: Result of IGD solving 15-objective DTLZ series

	ϵ	N	MOEA/D	DTAEA		ϵ	N	MOEA/D	DTAEA
DTLZ1	0.5	566	**3.46E-0(1.28E-0)**†	3.61E-1(3.34E-1)	DTLZ3	0.5	544	**4.01E-2(2.21E-2)**†	6.12E-1(0.21E-0)
	0.1	1230	1.42E-1(3.76E-2)	**2.21E-2(2.70E-1)**†		0.1	812	**2.21E-2(1.52E-2)**	3.11E-2(3.71E-1)
	0.01	3302	**2.17E-2(5.35E-3)**	2.24E-2(5.27E-2)		0.01	1808	1.31E-2(1.21E-3)	**5.12E-3(4.27E-3)**†
	0.001	-	-	-		0.001	-	-	-
DTLZ2	0.5	532	3.01E-2(2.21E-2)	**2.98E-2(0.21E-0)**	DTLZ4	0.5	882	3.51E-2(2.21E-0)	**2.45E-2(1.21E-1)**†
	0.1	722	1.76E-2(1.52E-2)	**1.54E-2(3.71E-1)**		0.1	1212	4.51E-2(2.47E-1)	**2.64E-2(5.23E-2)**†
	0.01	1210	1.44E-2(4.22E-3)	**2.01E-3(6.16E-3)**†		0.01	3808	2.31E-2(4.16E-2)	**6.32E-3(4.27E-2)**†
	0.001	8012	1.30E-2(8.62E-4)	**1.21E-3(2.36E-4)**†		0.001	7332	2.12E-2(2.25E-2)	**4.23E-3(3.25E-2)**†

Wilcoxon's rank sum test at a 0.05 significance level is performed between DTAEA and MOEA/D. † denote the performance of the corresponding algorithm is significantly better. The best median value is highlighted in boldface with gray background. '-' means that DTAEA cannot stop before 10000 generations.

For DTLZ1, DTAEA is worse than MOEA/D with $\epsilon = 0.5$ and $\epsilon = 0.01$. The reason that DTAEA has poor performance is because $\epsilon = 0.5$ is too large. DTAEA cannot achieve a good population before the objective is added. When $\epsilon = 0.01$, DTAEA waste too many function evaluations in the lower dimension problems. For example, DTAEA already has achieved a well-converged and well-distributed population for two-objective problems after about 200 generations. However, based on the ϵ, the DTAEA does not add any objective to the problem at that time, but runs for another 76 generations. This extra running is wasting computational resources, which makes DTAEA less effective than MOEA/D.

For DTLZ3, DTAEA is also worse than MOEA/D with $\epsilon = 0.5$ and $\epsilon = 0.1$. The reason is the same as the situation for DTLZ1. DTAEA cannot perform well without

enough number of runs. As the convergent speed in DTL3 is slower than the convergent speed in DTLZ1, when $\epsilon = 0.01$ DTAEA can win MOEA/D.

For DTLZ2 and DTLZ4, DTAEA wins in all the cases. It is interesting that the result from DTAEA running of 700 generations is better than the result from MOEA/D running for more than 1000 generations. One of the reason may be: comparing to DTLZ1 and DTLZ3, DTLZ2 and DTLZ4 are easier to converge. DTAEA easily achieves a converged population before every change.

This result is not very attractive as the drawbacks of using DTAEA is huge: 1) we do not find an appropriate method to decide when to add an objective. Methods based on Pre-setting work well, but request tuning many parameters. Adaptive methods cannot achieve a satisfying result, DTAEA do not obviously surpass the current algorithms. The conclusion for this research question is that unless a well-designed adaptive ϵ is added, it is hard to solve many-objective optimization problems in a changing number of objectives way. It is hard to say we can successfully use DTAEA to solve the many-objective problems.

3.5 Case Study: Scheduling Problems for Software Project Based on Rational Unified Process

Having tested DTAEA's ability in solving DMOPs with a changing number of objectives on test problems, this section tends to investigate the performance of DTAEA and the other peer algorithms on a case study about optimal the scheduling problem for software project based on Rational Unified Process (RUP). Plenty of models [126, 127] have been proposed to tackle the software project scheduling problem. RUP is one of the most popular life-cycle models, as shown in Figure 3.24. RUP has at least four phases, and each phases contains different processes (at most six), which are ongoing simultaneously. Employees require different salaries and have different skill levels. These employees can be scheduled to these processes in a phase. The Pareto-optimal set of scheduling are the

target of the optimization.

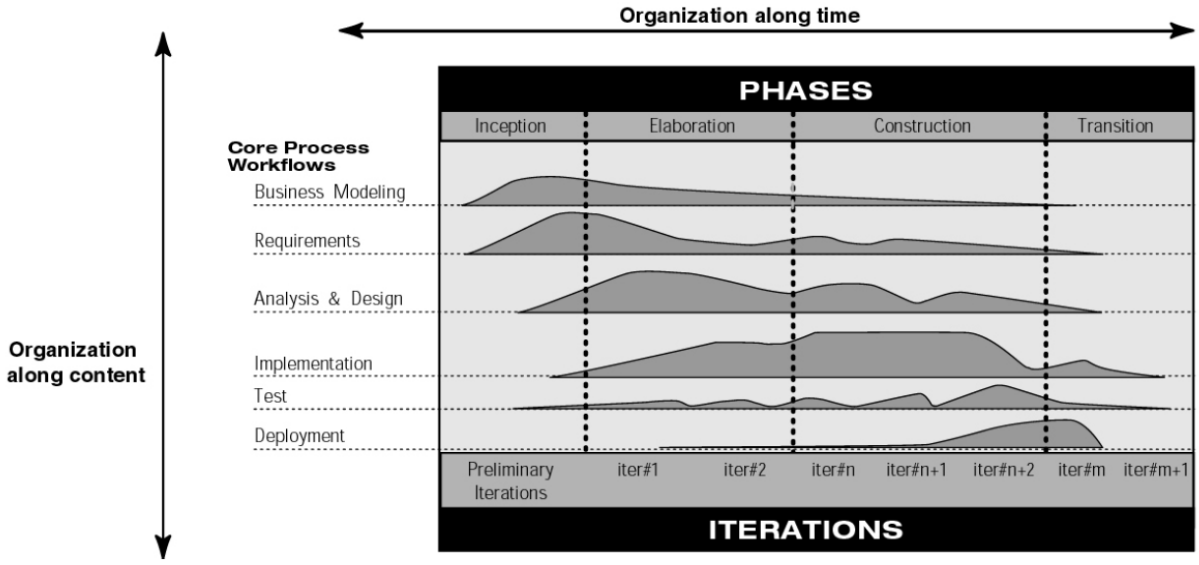


Figure 3.24: Flowchart of Rational Unified Process.

3.5.1 Model and the Objective Functions

Mod l

Here we first introduces the model for the scheduling problem on RUP, then we introduce the objectives. We assume there are e employees. For the ith employee (i=1,2,...,e), the attributes considered are as follows:

- p_i^n is the per-hour salary paid to employee who normally works for some tasks in the project.
- $\mathbf{S}_i = \{s_i^1, s_i^2, ..., s_i^K\}$ is the skill list for an employee. Where $s_i^j \in [0, 5]$. $s_i^j = 5$ means the ith employee masters skill j to highest level. $s_i^j = 0$ means employee ith does not have skill j. K is the total number of skills.

Assume that a software project has N phases, $\{t_1, t_2, ..., t_N\}, t_i \in 1, 2, 3, 4$. For each phase, there are 6 parallel processes executed. $P^t = \{p_1^t, ..., p_k^t\}, 1 \leq k \leq 6$. p_k^t is the kth process in the tth phase. The attributes considered for the process p_k^t are as follows:

- wr_k^t is the estimated work effort for the process.
- $\mathbf{Sr}_k^t = \{Sr_1, Sr_2, ...Sr_K\}$ is the required skill table for the process. The employees working on this process should meet all the requirement on this list.

The employee allocation matrix is:

$$\mathbf{W}^t = \begin{Bmatrix} w_{11}^t & \cdots & w_{1e}^t \\ \vdots & \vdots & \vdots \\ w_{61}^t & \cdots & w_{6e}^t \end{Bmatrix} \tag{3.21}$$

where w_{ij}^t indicates the number of hours for employee j working on process i in the phase t.

Objective: Salary

The objective of Salary which should be minimized is as follows:

$$Minimize \quad f_1(\mathbf{W}^t) = \sum_{i=1}^{i=6} \sum_{j=1}^{j=m} w_{ij} p_j^n \tag{3.22}$$

Objective: Duration Time

The duration time should be minimized:

$$Minimize \quad f_2(\mathbf{W}^t) = \max_{1 \leq i \leq 6} \frac{wr_i^t}{\sum_{j=1}^{j=m} w_{ij}} \tag{3.23}$$

Objectives: Quality for each Processes

Quality should be maximized for each process in every phase. The quality functions for each process can be calculated by COCOMO model [128].

Table 3.11: Median HV and iqr on the scheduling problem on RUP

phase	1	2	3	4
DTAEA	**0.721(1.23E-1)**	**0.823(5.31E-1)**	**0.733(3.54E-1)**	**0.434(3.34E-1)**
NSGA-II	0.121(2.37E0)	0.253(6.42E-1)	0.466(3.32E-1)	0.262(3.15E-1)
DNSGA-II	0.368(3.27E-1)	0.133(2.13E-2)	0.427(1.84E-1)	0.423(5.32E-1)
MOEA/D	0.511(5.21E-2)	0.324(2.66E-1)	0.6221(7.45E-1)	0.421(2.56E-1)
MOEA/D-KF	0.628(6.21E-2)	0.572(2.13E-2)	0.6267(2.81E-1)	0.423(3.12E-1)

3.5.2 Results

According to the Table 3.11, the DTAEA performs much better than other algorithms on a this simple scheduling problem, which have 10 employees and 4 phases. The results of other four algorithms do not even come close to DTAEA, which shows that DTAEA can handle DMOPs with a changing number of objectives in a satisfying way. It should be pointed out that on phase 4, the performance of other algorithms are close to the result of DTAEA. That is because one objective is added while two others are removed on phase 4. This situation is much more complicated than either increasing or decreasing objectives, which discussed in previous sections. It has to be admitted DTAEA cannot handle this situation very well. Even so, DTAEA still beats all other algorithms on the phase 4. All the data for calculation in this experiment can be found in Table 3.12 and Table 3.13.

Table 3.12: Skill table and the per hour salary for 10 employees

id	s^1	s^2	s^3	s^4	s^5	s^6	p^n	id	s^1	s^2	s^3	s^4	s^5	s^6	p^n
1	2	1	0	1	4	4	16.4	16	4	2	4	1	4	2	33.4
2	2	2	5	2	2	0	21.2	17	3	2	1	4	1	5	31.8
3	1	3	3	4	3	2	27.8	18	4	2	5	3	2	1	20.1
4	4	3	3	1	4	2	25.1	19	1	2	4	0	2	1	12.3
5	2	2	5	3	3	4	34.3	20	4	5	3	1	1	3	21.7
6	2	3	1	1	2	5	20.5	21	0	4	3	4	3	5	25.1
7	3	1	2	4	3	3	17.6	22	4	0	3	2	4	3	29.8
8	2	5	1	3	2	0	15.1	23	4	2	4	2	3	4	36.9
9	3	4	5	4	2	3	37.5	24	1	5	2	1	0	3	12.6
10	0	3	1	4	5	3	23.3	25	5	5	1	1	4	4	25.7

Table 3.13: Workloads for the Scheduling Problem

Phase/Process	1	2	3	4
Modelling	4	13	0	0
Requirements	16	31	0	0
Design	8	62	87	0
testing	0	22	186	16
Assessment	0	17	131	20
Deployment	0	0	0	26

3.6 Conclusion

While the current studies on dynamic multi-objective optimization focus on problems with time-dependent objective functions, this chapter distinguishes itself by working on the DMOPs with a time varying number of objectives. This type of dynamics leads to the expansion or contraction of the PF or PS manifold with the increase or decrease of the number of objectives. Hence the existing dynamic handling techniques are not applicable to this scenario. To address this problem, dynamic two-archive EA (denoted as DTAEA) is proposed to process DMOPs with a changing number of objectives. DTAEA simultaneously maintains two co-evolving populations during the evolution process, i.e. CA and DA. They are complementary: the CA takes care more about the convergence while the DA concerns more about the diversity. The CA and the DA are separately reconstructed whenever the environment changes. In the meantime, they are maintained differently. Using a restricted mating selection mechanism, DTAEA takes advantages of the complementary effects of CA and DA, striking the balance between convergence and diversity at all times. Comprehensive experiments on a set of dynamic multi-objective benchmark problems fully demonstrate the superiority of DTAEA in tackling problems with a dynamically changing number of objectives. In particular, it is able to effectively and efficiently track the expansion or contraction of the PF or PS manifold as the environment changes. In the experiments, some state-of-the-art dynamic EMO algorithms surprisingly show inferior performance than their stationary counterparts.

DTAEA is only the first attempt towards a systematic investigation of the methods for handling the DMOPs with changing number of objectives. The purpose of this chapter

is to demonstrate that DMOPs with a changing number of objectives is challenging, and methods such as DTAEA can solve the problem better than current algorithms. Further study invites more attention and efforts. The algorithms design is awaiting in further research, as DTAEA actually only reacts towards changing number of objectives but ignores those with moving PS. Adding mechanism into DTAEA obviously can improve the performance of DTAEA. Future work can exploit the dynamic environments with inconspicuous and low-frequency changes. Also, the research questions in Section 3.4 can inspire more interesting answers. For example, the research question seven can become a brand-new idea to solve many-objective problems. In addition, the benchmark problems used in this chapter are developed from the classic DTLZ benchmark suite. It is a good attempt to develop more challenging benchmark problems with other characteristics, e.g., from the WFG [65] and the CEC 2009 competition [85] benchmark suites.

It should be pointed out that TAEA is not the only method for solving DMOPs with a changing number of objectives. TAEA is a possible method proposed in this chapter. The reason for choosing TAEA is: different strategies (selection, reconstruction, reproduction) can be developed for the separated two archives. TAEA can handle dynamics by the help of these archives. There are plenty of possible methods. For example, current predicted-based algorithms only predict the changing shape or movement of the PS/PF. Obvious it is possible to predict the PS/PF's manifold based on mathematical modelling such as Radial Basis Function network.

CHAPTER 4

TWO-ARCHIVE EVOLUTIONARY ALGORITHM FOR DYNAMIC CONSTRAINED MULTI-OBJECTIVE OPTIMIZATION

Solving constrained multi-objective optimization problems (CMOPs) has attracted many researchers. In the past decades, many algorithms have been proposed to solve CMOPs [11, 32, 48, 67–70]. Most, if not all, of the existing constraint handling techniques tend to push a population toward the feasible areas as much as possible before considering the balance between convergence and diversity within the feasible region. Few works have paid attention to exploring the infeasible areas. However, in some complex situations, especially when the constraints change during the optimization process, algorithms cannot handle the constrained problems without exploring the infeasible area. In this chapter, we first discuss some complex situations for the static constrained multi-objective problems, then we investigate the dynamic constrained multi-objective problems (DCMOPs). These problems show the importance of exploring the infeasible areas. To address these constrained problems, we propose a parameter-free constraint handling technique based on a two-archive evolutionary algorithm, denoted as C-TAEA. In particular, to complement the behaviour of the convergence archive and provide as much diversified information as possible, the diversity archive aims at exploring areas under-exploited by the convergence archive including the infeasible regions.

This chapter is organised in the following way. Section 4.1 clarifies the motivation.

Section 4.2 is the implementation of the algorithms. Section 4.3 shows the results of the experiments. Section 4.4 is a case study from water distribution network problems. Section 4.5 discusses problems with a changing number of objectives and dynamic constraints.

4.1 Motivation

Many works have put efforts on constraints handling techniques which have already been discussed in Section 2.3. However, most, if not all, of the existing constraint handling techniques always prefer feasible solutions than infeasible ones. Searching the infeasible regions is considered as a waste of computational resource. However, the overemphasis on the importance of the feasibility can lead to an ineffective search when the algorithms encounter complex constraints, which will be shown in the following part. The dynamic constraints multi-objective problems mentioned here only involve the changing constrained functions.

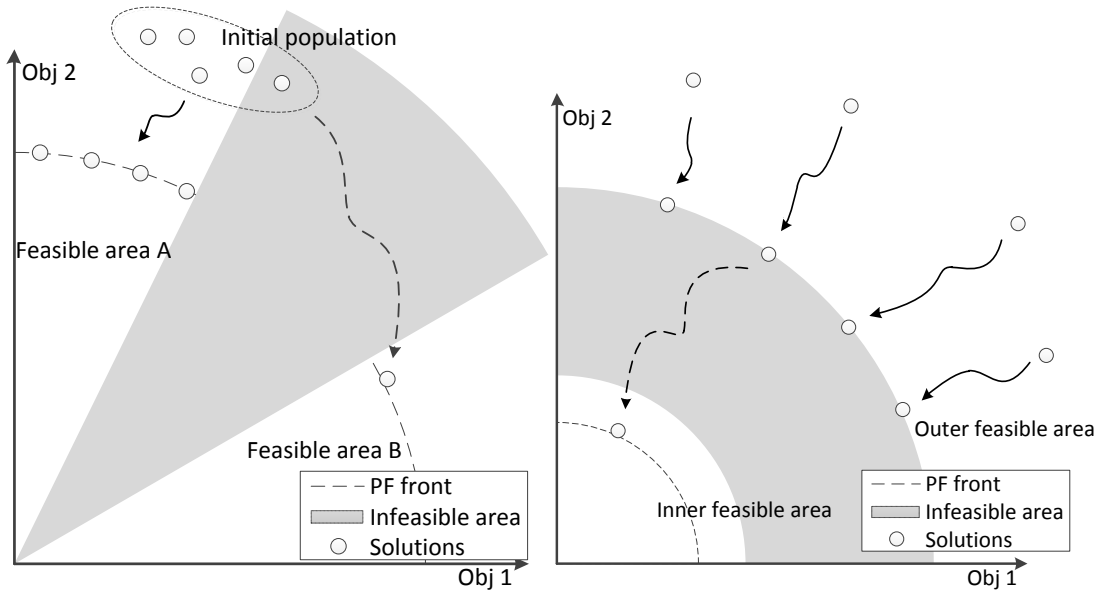
4.1.1 Challenges for Dynamic Constraints

Although single objective optimization problems with dynamic constraints are widely discussed, the multi-objective problems with dynamic constraints are nearly untouched. It should be noted that dynamic constraints multi-objective problems mentioned here only involve the changing constrained functions. The two types of constraints we discuss here are 1) Problems with partitioned objective space, 2) Problems with local optimums for constraints.

Problems with partitioned feasible region

The objective space (search space) of the problem sometimes is partitioned into several disconnected feasible regions by constraints. Figure 4.1 shows two examples of problems

with partitioned objective space.



(a) Example 1 (b) Example 2

Figure 4.1: Examples of problems with partitioned objective space

In Figure 4.1(a), the objective space is partitioned into feasible region A and B. Both regions cover part of the PF. Although we hope that the initial population is well distributed in the objective space, usually it is just an ideal assumption. In problems with strong bias such as C1-DTLZ3 test problem in the constrained DTLZ series [64], the initial population is usually distributed in a small region. When this happens as shown in Figure 4.1(a), solutions can easily converge to the PF in the nearby feasible area A (solid line arrow), while it is very difficult to reach the PF in the feasible area B (dashed line arrow). This makes it very hard for the algorithms to find the whole PF.

In Figure 4.1(b), the objective space is partitioned into the inner feasible region and the outer feasible region. The whole PF is in the inner region. If the search space is huge, the algorithms fail to generate any initial solutions inside the inner feasible region. If all the initial solutions are generated in the outer feasible region, algorithms can push the solutions towards the boundary of the outer feasible area (solid line arrow), but nearly impossible to push the solutions going through the infeasible rings between the inner and outer feasible region (dashed line arrow). This makes it very difficult for the algorithms

to converge to the PF.

When the constraints change during the optimization process, some of the infeasible solutions found by the algorithms may be more valuable than feasible ones. This may happen when: 1) these infeasible solutions become feasible after the changes, or 2) these infeasible solutions help the algorithms to track the PF after the changes. Two examples here show the importance of maintaining infeasible solutions when handling dynamic constraints. The first example is shown in Figure 4.2(a). Solution #1 in the figure is an

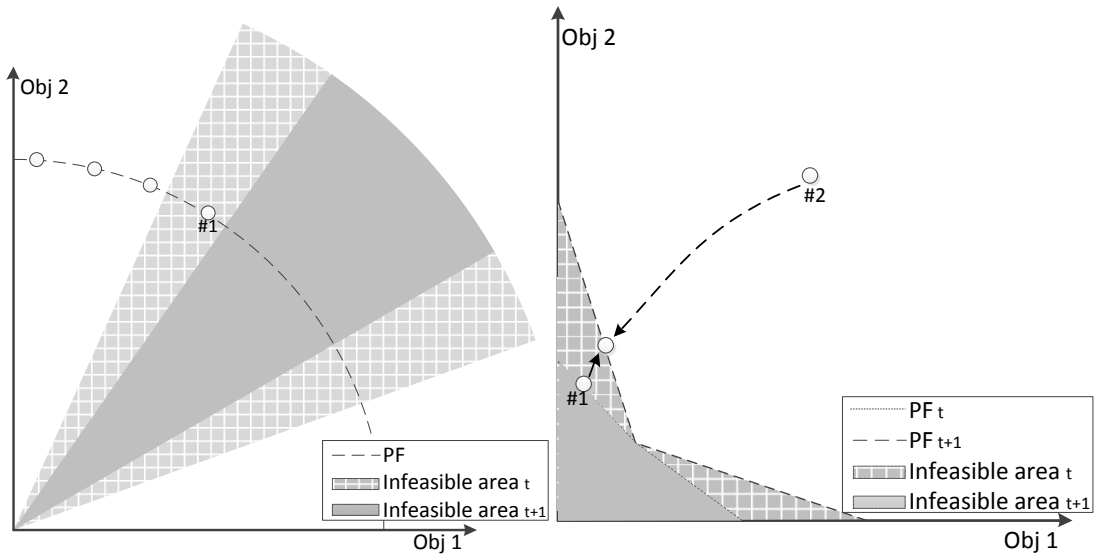


Figure 4.2: Examples for problems with dynamic constraints

infeasible solution at time t and becomes feasible at time $t+1$. If we knew this, discarding Solution #1 would not be a good decision at time t. The second example is based on the two-objective dynamic C3-DTLZ1 test problem. The objective functions for C3-DTLZ1 are the same as those for DTLZ1 (the dynamic C-DTLZ test suite will be discussed in detail in Section4.3.1).

$$\begin{aligned} minimize \quad & f_1 = (1+g)0.5\textstyle\prod_{i=1}^{m-1} x_i) \\ & f_{j=2:m-1} = (1+g)0.5(\textstyle\prod_{i=1}^{m-j} x_i)(1-x_{m-j+1}) \\ & f_m = (1+g)0.5(1-x_1) \end{aligned} \tag{4.1}$$

while the constraints are:

$$s.t. \quad c_j(\mathbf{x}) = \sum_{i=1, i\neq j}^{m} f_j(\mathbf{x}) + \frac{f_i(\mathbf{x})}{0.5+0.25sin(a\pi t)} - 1 \geq 0, \forall j = 1, ..., m \tag{4.2}$$

The result is shown in Figure 4.2(b). We assume that at time t there is a feasible Solution #1 near the PF. After the changes, Solution #1 becomes infeasible. To track the PF after the changes, it is easier to use Solution #1 as parents than to use some randomly generated feasible solutions (such as Solution #2).

The discussion above shows that it is worthwhile to explore the infeasible regions and maintain infeasible solutions. However, current research overemphasises the feasibility of the solutions, with limited attention to the infeasible solutions.

Problems with local optimum for constraint violation

Most methods are based on the value of constraint violation(CV). The value of constraint violation for a solution usually indicates the "distance" from the solution to the closest feasible area. Feasible solutions have $CV = 0$. The closer that an infeasible solution moves to feasible area, the smaller CV value it has. The existing research proposed many "distance" measurements. The well-known one proposed by Deb et al. [64] is defined as:

$$CV(\mathbf{x}) = \sum_{j=1}^{p+q} c_j(\mathbf{x}) \tag{4.3}$$

c_j is the degree of j-th constraint:

$$c_j = \begin{cases} |\min(G_j(\mathbf{x}), 0)| & j \leq p \\ |H_{j-p}(\mathbf{x})| & j > p \end{cases} \tag{4.4}$$

If $CV(\mathbf{x}) = 0$, $\mathbf{x}$ is a feasible solution, otherwise $\mathbf{x}$ is an infeasible solution.

However, the "distance" measurements may fail when there is local optimum for the constraint violation. Take the DC2-DTLZ2 as an example (the DC-DTLZ test suite will

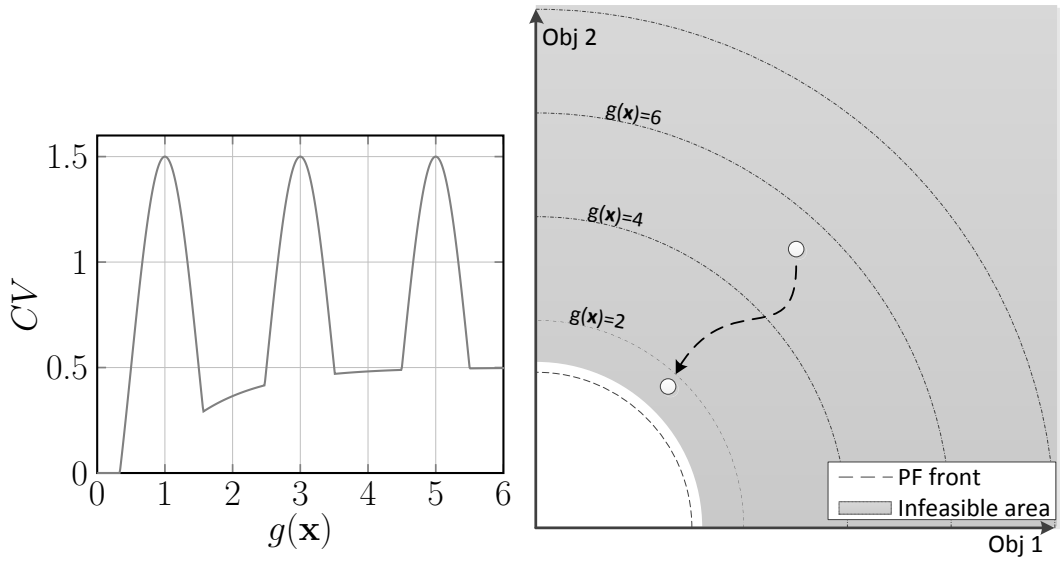
be discussed in detail in Section 4.3.1). The objective functions of DC2-DTLZ2 is the same as DTLZ2.

$$\begin{aligned} minimize \quad & f_1 = (1+g)0.5\prod_{i=1}^{m-1}\cos(x_i\pi/2) \\ & f_{j=2:m-1} = (1+g)0.5(\prod_{i=1}^{m-j}\cos(x_i\pi/2))(\sin(x_{m-j+1}\pi/2)) \\ & f_m = (1+g)\sin(x_1\pi/2) \end{aligned} \tag{4.5}$$

while the constraints are:

$$\begin{aligned} s.t. \quad & \cos(a\pi g) > b \\ & e^{-g} > b \end{aligned} \tag{4.6}$$

in which $g = \sum_{i=m}^{n}(x_i - 0.5)^2$ and a, b are parameters for the test problem. In this example, $a = 1, b = 0.5$. Figure 4.3 shows the CV calculated according to Equation 4.4. From this figure, we can see the CV does not always decrease when solutions move closer



(a) polt of $g(\mathbf{x}) - CV$ (b) objective space

Figure 4.3: CV for DC2-DTLZ2

to the feasible area (dashed line arrow). There are multiple local optimum for the CV. Solutions are easily trapped in the local CV optimum as the algorithms always prefer solutions with smaller CV. The algorithms will not push solutions out from the local CV optimum (dashed line arrow). The situation becomes more challenging when the local

CV optimum move over time. The algorithms will put all the affords tracking the local CV optimum, instead of pushing solutions towards the feasible regions.

4.2 Implementation

In this section, the DTAEA is adapted for the constrained problems, named as constrained TAEA (C-TAEA). The framework of C-TAEA is similar to DTAEA, but the mechanisms in the algorithms are different.

4.2.1 Reconstruction Mechanism

The basic idea of reconstruction is relocating solutions in CA and DA. Reconstruction moves all feasible solutions in DA to CA while move all the infeasible solutions in CA to DA. When the number of solutions in CA or DA is surpass the population size, the selection mechanism is used to remove solutions until the size of CA or DA equal to the population size. If the number of solutions in CA or DA is fewer than the population size, the CA or DA is filled up with mutated solutions from the solutions in CA or DA. The algorithm is described in Algorithm 9.

The *FeasibleSolutionSelection* picks all the feasible solutions from the set, and *InfeasibleSolutionSelection* picks all the infeasible ones. The Binary Tournament selection is described in Algorithm 10.

4.2.2 Update Mechanisms

Update Mechanism of the CA

The CA prefers those feasible non-dominated solutions, or the solutions with less CV value if feasible solutions are not enough. At the beginning, all feasible solutions are selected and put into the CA. If $|CA| > N$, the update mechanism for CA is the same as DTAEA. Otherwise, infeasible solutions with smaller CV value will be picked first. This

Algorithm 9: Reconstruction Mechanism

Input: The CA,DA before changes, population size N
Output: CA, DA

```
P ← CA ⋃ DA;
CA ← FeasibleSolutionSelection(P);
CA ← FeasibleSolutionSelection(P);
if |CA| > N then
    updateCA(CA);                                   // update CA as DTAEA
if |DA| > N then
    CVSection(DA);                  // choose solutions with smaller CV
while |DA| < N do
    x^c ← BinaryTournamentSelection(DA);
    x^m ← PolynomialMutation(x^c);
    DA ← DA ⋃ {x^m};
while |CA| < N do
    x^c ← BinaryTournamentSelection(CA);
    x^m ← PolynomialMutation(x^c);
    CA ← CA ⋃ {x^m};
return CA and DA
```

Algorithm 10: Binary Tournament Selection

Input: Solution set S
Output: Selected solution $\mathbf{x}^c$

```
Randomly select two solutions x^1 and x^2 from S;
if Density(x^1) < Density(x^2) then
    x^c ← x^1;
else if Density(x^1) > Density(x^2) then
    x^c ← x^2;
else
    x^c ← Randomly pick one between x^1 and x^2;
return x^c
```

action will repeat until the CA is full. The pseudo-code of the update mechanism of the CA is given in Algorithm 11.

Algorithm 11: Update Mechanism of the CA

Input: CA, offspring population Q, weight vector set W
Output: Updated CA

```
1  S ← ∅, i ← 1;
2  R ← CA ⋃ Q;
3  P_f ← FeasibleSolutionSelection(R);     // choose all feasible solutions
4  P_i ← (R − P_f);
5  if |P_f| ≤ N then
6      updateCA(P_f);                          // update CA as DTAEA with P_f
7  else
8      CA ← P_f;
9      while |CA| ≤ N do
10         p ← CVSection(P_i);    // choose solutions with smaller value of CV
11         CA ← {p} ⋃ CA;
12         P_i ← P_i \ p;
13 return CA
```

Update Mechanism of the DA

DA keeps solutions that 1) improve diversity but are dominated by other solutions, 2) are infeasible but dominates the feasible ones. The dominate relation mention here is the original dominance relation, instead of the constrained dominance relation (CDR). According to these two conditions, the update mechanism of the DA is designed. The pseudo-code of the update mechanism of the DA is given in Algorithm 12.

4.2.3 Offspring Reproduction

The mating mechanism is the most important part in the C-TAEA. There are two conditions where parents should be chosen from DA.

- The CA is well-converged, while the diversity is unsatisfied.

Algorithm 12: Update Mechanism of the DA

Input: CA, DA, offspring population Q, weight vector set W
Output: Updated DA

```
S ← ∅, i ← 1;
R ← DA ⋃ Q;
{Δ^1(t), ··· , Δ^|W|(t)} ← Association(R, W);
itr ← 1;
while |S| ≤ N do
    for i ← 1 to |W| do
        Flag ← False;
        foreach x^j ← 1 ∈ |Δ^i(t)| do
            if Dominate(s_j, CA) then ; // If s_j dominates all solutions in CA
                Δ^i(t) ← Δ^i(t) \ {x^j};
                S ← S ⋃ {x^j};
                Flag ← True;
                Break;     // add s_j into DA, and finish the loop for Δ^i(t)
        if Flag ∧ Δ^i(t) ≠ ∅ ∧ CA has less than itr solutions in Δ^i(t) then
            O ← NonDominationSelection(Δ^i(t));
            x^b ← argmin_{x∈O} {g^tch(x|w^c(t), z*(t))};
            Δ^i(t) ← Δ^i(t) \ {x^b};
            S ← S ⋃ {x^b};
    itr ← itr + 1;
DA ← S;
return DA
```

- The CA is well-converged, while some solutions from DA dominates all solutions from CA.

According to this two conditions, a dominated rate I_d is calculated. The dominated rate is defined as:

$$I_d = \frac{|DA \wedge \texttt{NonDominationSelection}(DA \bigcup CA)|}{|\texttt{NonDominationSelection}(DA \bigcup CA)|} \tag{4.7}$$

The pseudo-code of the mating mechanism is given in Algorithm 13. The `NonDominationSelection` selects all the non-dominated solutions from the input.

Algorithm 13: Mating Selection Mechanism

Input: CA, DA
Output: Mating parents $\mathbf{p}_1$, $\mathbf{p}_2$

```
if rnd > I^d then
    p_1 ← RandomSelection(CA);
else
    p_1 ← RandomSelection(DA);
if rnd < I^o_CA then
    p_2 ← RandomSelection(CA);
else
    p_2 ← RandomSelection(DA);
return p_1, p_2
```

4.3 Results

In this section, we discuss the empirical results on different benchmark problems separately.

4.3.1 Benchmark Problems

In this section, we introduce the widely used C-DTLZ series from Deb. et al. [64], then we propose our test problems for static constrained multi-objective problems, which are denoted as DC-DTLZ series. In the end, we extend C-DTLZ series to a dynamic version, denoted as dynamic C-DTLZ series.

Table 4.1: Mathematical Definitions of DTLZ Test Problems

Problem Instance	Definition	Domain
DTLZ1	$f_1 = (1+g)0.5\prod_{i=1}^{m-1} x_i$ $f_{j=2:m(t)-1} = (1+g)0.5(\prod_{i=1}^{m-j} x_i)(1-x_{m-j+1})$ $f_m = (1+g)0.5(1-x_1)$ $g = 100[n-m+1+\sum_{i=m}^{n}((x_i-0.5)^2-\cos(20\pi(x_i-0.5)))]$	$[0,1]$
DTLZ2	$f_1 = (1+g)0.5\prod_{i=1}^{m-1}\cos(x_i\pi/2)$ $f_{j=2:m-1} = (1+g)0.5(\prod_{i=1}^{m-j}\cos(x_i\pi/2))(\sin(x_{m-j+1}\pi/2))$ $f_m = (1+g)\sin(x_1\pi/2)$ $g = \sum_{i=m}^{n}(x_i-0.5)^2$	$[0,1]$
DTLZ3	as DTLZ2, except g is replaced by the one from F1	$[0,1]$
DTLZ4	as DTLZ2, except x_i is replaced by x_i^α, where $i \in \{1,\cdots,m-1\}, \alpha > 0$	$[0,1]$

C-DTLZ Benchmark Suite

The existing test problems include C-DTLZ series, CF series and the CTP series. The C-DTLZ series are the only test suite that is scalable. The C-DTLZ series are based on the DTLZ test suite. The DTLZ test suite is listed in Table 4.1.

The constrained C-DTLZ series add constraints to the DTLZ series. All the objective functions in C-DTLZ series are the same as DTLZ series, while the constraint for C1-DTLZ1 is:

$$c(\mathbf{x}) = 1 - \frac{f_m(\mathbf{x})}{0.6} - \sum_{i=1}^{m}(m - \frac{f_i(\mathbf{x})}{0.5}) \geq 0 \tag{4.8}$$

The C1-DTLZ1 has a small feasible region, while the major part of the search space is infeasible. However, this feature actually does not cause much difficulty for algorithms to handle the constraint. This is because the constraint in this problem is linear. That means a solution has a smaller CV when it is closer to the PF. Any CV-driven algorithms can easily handle this problem.

The constraint for C1-DTLZ3 is:

$$c(\mathbf{x}) = (\sum_{i=1}^{m} f_i(\mathbf{x})^2 - 16)(\sum_{i=1}^{m} f_i(\mathbf{x})^2 - r^2) \geq 0. \tag{4.9}$$

The C1-DTLZ3 is the most challenging one in the C-DTLZ series. As the search space is huge, the initial solutions of any algorithms usually is not located in the inner feasible

area. As most of the current algorithms are CV-driven, algorithms will not attempt to push solutions into the infeasible barrier. This makes most of the algorithms fail to converge to the PF.

The constraint for C2-DTLZ2 is:

$$c(\mathbf{x}) = \max\{\max_{i=1}^{m}[(f_i(\mathbf{x})-1)^2 + \sum_{j=1,j\neq i}^{m}(f_j^2 - r^2)], [\sum_{i=1}^{m}(f_i(\mathbf{x}) - \frac{1}{\sqrt{m}})^2 - r^2]\} \quad (4.10)$$

The difficulty of C2-DTLZ2 is adjustable. The C2-DTLZ2 has several disconnected part of PF. The parameter r in C2-DTLZ2 impacts the size of the feasible areas. The larger r is, the larger feasible areas are. When the r in C2-DTLZ2 is large, e.g. $r > 3$, this problem can be handled by most of the algorithms. When we set a smaller r, e.g. $r = 0.5$, many algorithms fail to find all the part of the Pareto-optmal front. The constraints for C3-DTLZ1 are:

$$c_j(\mathbf{x}) = \sum_{i=1,i\neq j}^{m} f_j(\mathbf{x}) + \frac{f_i(\mathbf{x})}{0.5} - 1 \geq 0, \forall_j = 1, ..., m. \quad (4.11)$$

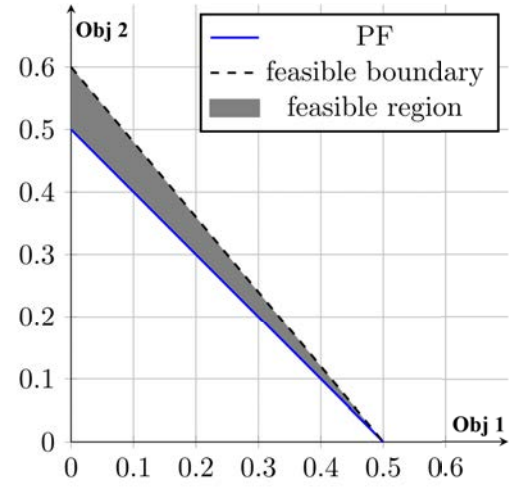
The constraints for C3-DTLZ4 are:

$$c_j(\mathbf{x}) = \frac{f_j^2}{4} + \sum_{i=1,i\neq j}^{m} f_i(\mathbf{x})^2 - 1 \geq 0, \forall_j = 1, ..., m. \quad (4.12)$$

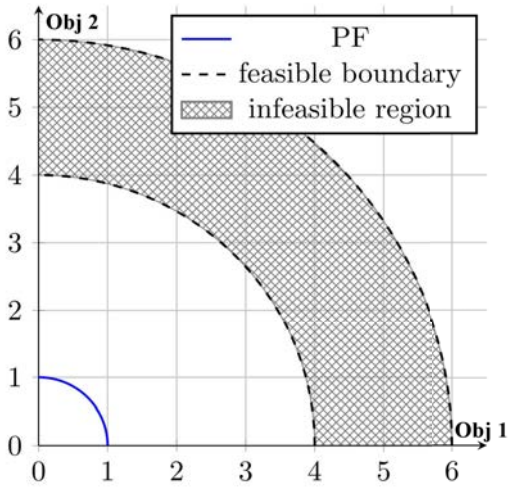
The C3-DTLZ1 and C3-DTLZ4 bring no difficulties at all. Any solutions preferring feasible solutions to infeasible ones can handle them. The 2 objective instances for these problems is shown in Figure 4.4.

DC-DTLZ Benchmark Suite

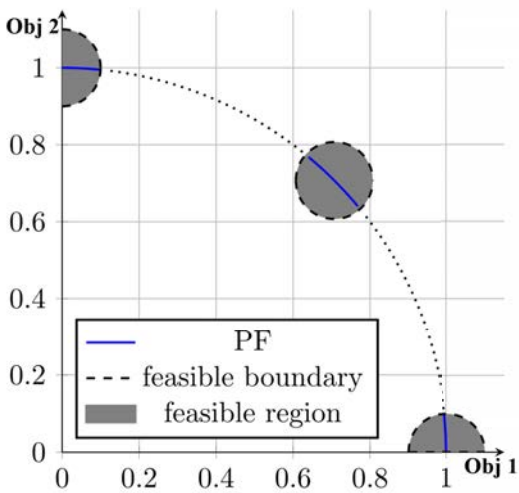
As C-DTLZ benchmark suite does not cover all the challenges mention in Section 4.1, here we propose our test benchmark suite, also extended from DTLZ series, denoted as DC-DTLZ series. All the objective functions in DC-DTLZ test suite are the same as DTLZ



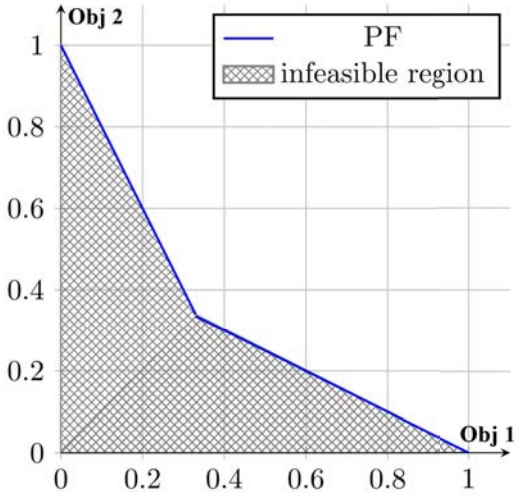
(a) C1-DTLZ1



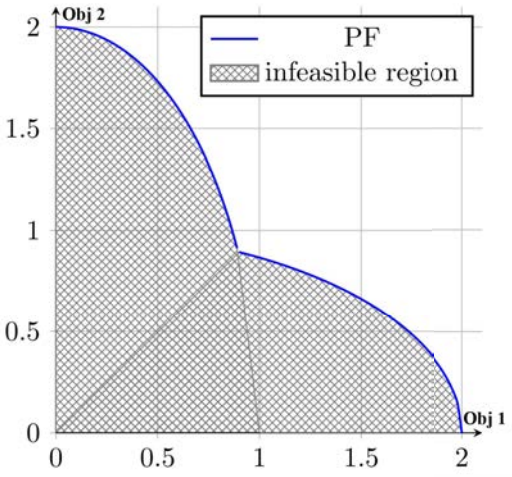
(b) C1-DTLZ3



(c) C2-DTLZ2



(d) C3-DTLZ1



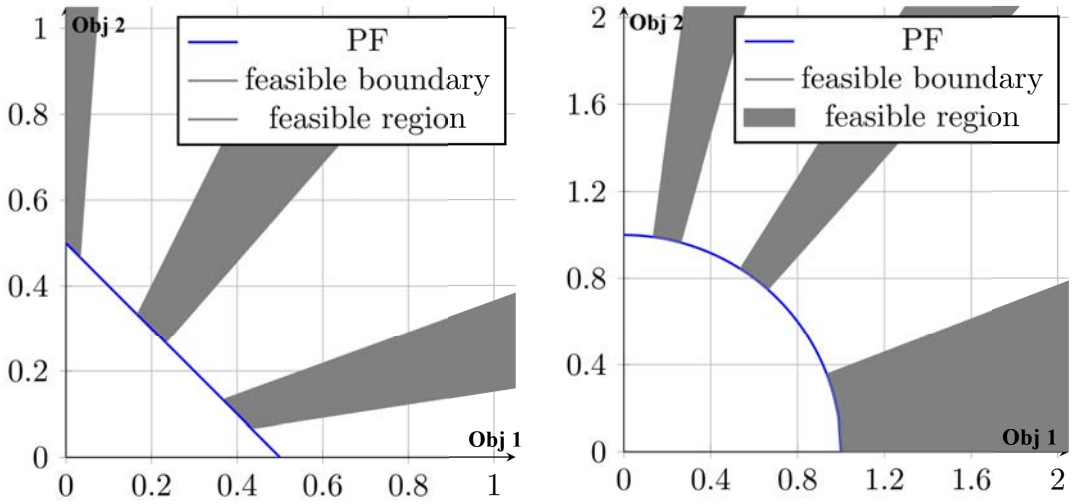
(e) C3-DTLZ4

Figure 4.4: Illustration of C-DTLZ Series in 2D

series, with additional constraints. The constraint for DC1-DTLZ1 and DC1-DTLZ3 is:

$$c(\mathbf{x}) = cos(a\pi x_1) > b \tag{4.13}$$

There are two parameters a and b for DC1-DTLZ1 and DC1-DTLZ3. $a > 0$ controls the number of feasible areas on the PF. Each feasible area is a cone starting from the origin, as shown in Figure 4.5. $0 < b < 1$ controls the size of each feasible area, where a large b leads to a narrow feasible segment. In our empirical studies, we suggest $a = 3$ and $b = 0.5$.



(a) DC1-DTLZ1 (b) DC1-DTLZ3

Figure 4.5: Illustration of DC1-DTLZ1 and DC1-DTLZ3 in 2D

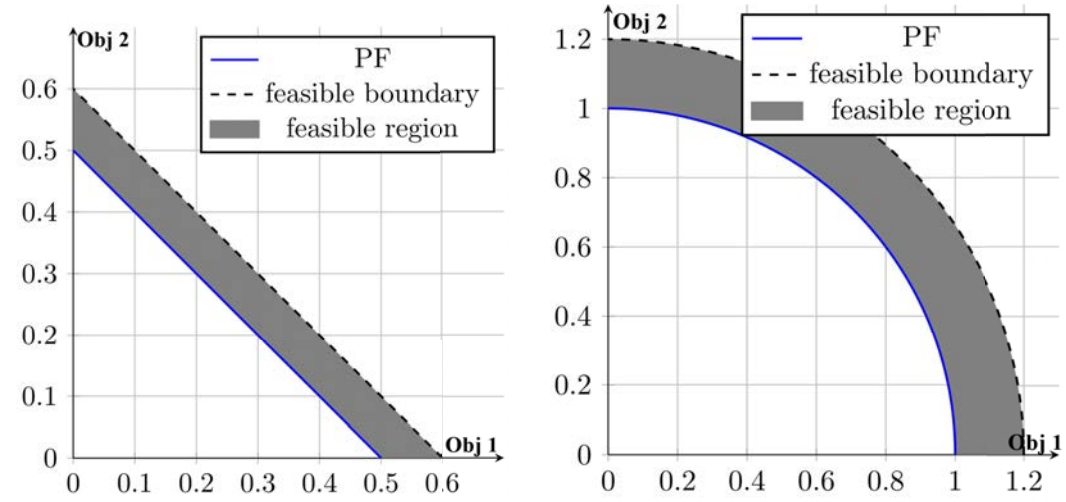
The constraints for DC2-DTLZ1 and DC2-DTLZ3 are:

$$c_1(\mathbf{x}) = cos(a\pi g(\mathbf{x}_m)) > b \tag{4.14}$$

$$c_2(\mathbf{x}) = e^{-g(\mathbf{x}_m)} > b \tag{4.15}$$

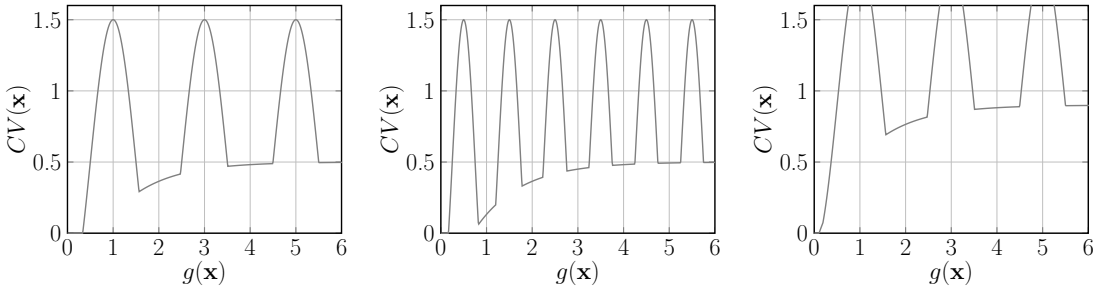
The DC2-DTLZ1 and DC2-DTLZ3 have a similar infeasible area as C1-DTLZ1 as shown in Figure 4.6. However, DC2-DTLZ1 and DC2-DTLZ2 are much more difficult to tackle. In DC2-DTLZ1 and DC2-DTLZ3, CV for a solution does not monotonically decrease

when it converges toward the PF, while there exists several local optima of CV which can be found in the Figure 4.7. $a > 0$ controls the number of local optimums of CV, while $0.5 < b < 1$ controls the value of the corresponding local optimum.



(a) DC2-DTLZ1 (b) DC2-DTLZ3

Figure 4.6: Illustration of DC2-DTLZ1 and DC2-DTLZ3 in 2D



(a) $a = 1, b = 0.5$ (b) $a = 2, b = 0.5$ (c) $a = 1, b = 0.9$

Figure 4.7: Variation of $CV(\mathbf{x})$ with respect to $g(\mathbf{x})$

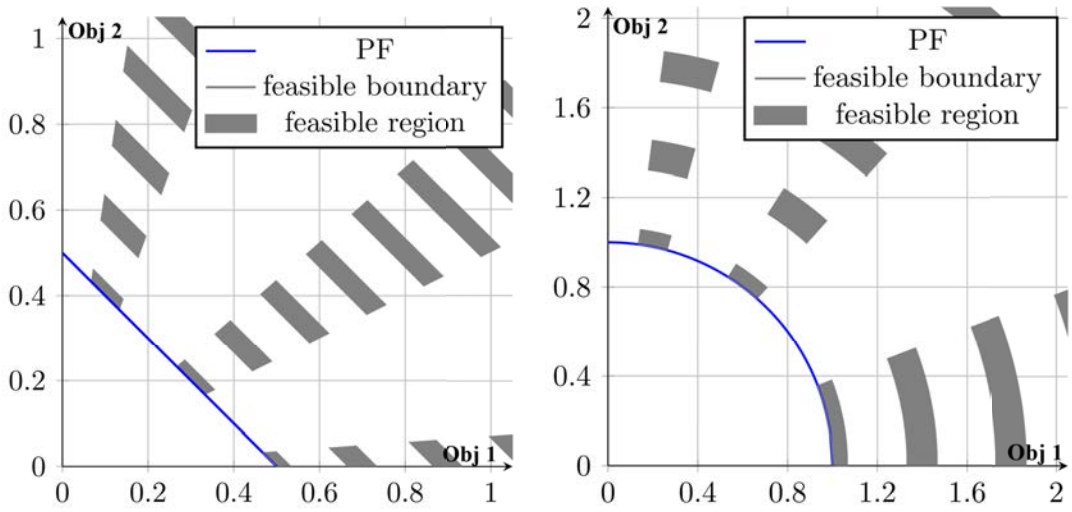
The constraints for DC3-DTLZ1 and DC3-DTLZ3 are:

$$c_j(\mathbf{x}) = cos(a\pi x_j) > b, \forall_j = 1, ..., m \tag{4.16}$$

$$c_{m+1}(\mathbf{x}) = cos(a\pi g(\mathbf{x}_m)) > b. \tag{4.17}$$

Similar to DC1-DTLZ1, DC3-DTLZ1 and DC3-DTLZ3 have a partitioned objective space. Many disconnected feasible areas are separated by the infeasible areas, as shown in Figure

4.8. The DC3-DTLZ1 and DC3-DTLZ3 also hard to handle, as solutions are easily trapped in the feasible areas and fail to converge towards the PF.



(a) DC2-DTLZ1 (b) DC2-DTLZ3

Figure 4.8: Illustration of DC2-DTLZ1 and DC2-DTLZ3 in 2D

Dynamic C-DTLZ Benchmark Suite

Here we propose a test problems set with dynamic constraints functions. The number of constraints is not changed during the optimization process. In all the problems, we use time point $t = \lfloor \frac{\tau}{\tau_t} \rfloor$, τ and $\frac{1}{\tau_t}$ represent the iteration counter and the frequency of change. All the problems have the same objective functions as the original problems from DTLZ series with additional constraints.

The constraint for dynamic C1-DTLZ3 is:

$$c(\mathbf{x}) = (\sum_{i=1}^{m} f_i(\mathbf{x})^2 - 16)(\sum_{i=1}^{m} f_i(\mathbf{x})^2 - r^2) \geq 0. \tag{4.18}$$

in which $r = 8 + 2\cos(0.25\pi t)$. The dynamic C1-DTLZ3 also has an infeasible barrier between the inner feasible area and the outer feasible area. However, the size of the infeasible barrier is changing. The barrier range change from $[4, 6]$ (inner boundary) to $[4, 10]$ (outer boundary). Compare to the static version, the dynamic C1-DTLZ3 is

more challenging, as the changing size of infeasible area makes it hard to converge to the boundary of the outer feasible area, not to mention converging to the PF.

The constraint for dynamic C2-DTLZ2 is:

$$c(\mathbf{x}) = \max\{\max_{i=1}^{m}[(f_i(\mathbf{x})-1)^2 + \sum_{j=1,j\neq i}^{m}(f_j^2 - r^2)], [\sum_{i=1}^{m}(f_i(\mathbf{x}) - r_i)^2 - r^2]\} \tag{4.19}$$

in which $r_i = \frac{\cos(0.25\pi(t+i)}{\sqrt{m}}$. The feasible areas in C2-DTLZ2 is similar to the static version, except that the feasible area in the centre of the PF is circling around the centre point. This problem gives the challenge in tracing the new feasible area.

The constraints for dynamic C3-DTLZ1 are:

$$c_j(\mathbf{x}) = \sum_{i=1,i\neq j}^{m} f_j(\mathbf{x}) + \frac{f_i(\mathbf{x})}{r} - 1 \geq 0, \forall_j = 1, ..., m. \tag{4.20}$$

in which $r = 1 + 0.5\cos(0.25\pi t)$.

The constraints for dynamic C3-DTLZ4 are:

$$c_j(\mathbf{x}) = \frac{f_j^2}{4} + \sum_{i=1,i\neq j}^{m} f_i(\mathbf{x})^2 - 1 \geq 0, \forall_j = 1, ..., m. \tag{4.21}$$

in which $r = 4(1 + 0.5\cos(0.25\pi t))$.

The feasible area is changing its shape during the optimal process, which makes the PF move. The challenge for this test problem is tracing the moving PF.

Dynamic DC-DTLZ Benchmark Suite

In this part, we propose our dynamic DC-DTLZ benchmark suite. All the parameter such as a and b is the same as the static version.

The constraint for dynamic DC1-DTLZ1 and dynamic DC1-DTLZ3 is:

$$c(\mathbf{x}) = cos(a\pi t x_1) > b \tag{4.22}$$

The constraints for dynamic DC2-DTLZ1 and dynamic DC2-DTLZ3 are:

$$c_1(\mathbf{x}) = cos(a\pi tg(\mathbf{x}_m)) > b \tag{4.23}$$

$$c_2(\mathbf{x}) = e^{-g(\mathbf{x}_m)t} > b \tag{4.24}$$

The constraints for dynamic DC3-DTLZ1 and dynamic DC3-DTLZ3 are:

$$c_j(\mathbf{x}) = cos(a\pi tx_j) > b, \forall_j = 1, ..., m \tag{4.25}$$

$$c_{m+1}(\mathbf{x}) = cos(a\pi tg(\mathbf{x}_m)) > b. \tag{4.26}$$

4.3.2 Performance Metrics

We use the standard IGD and HV as the performance metrics.

The IGD indicator calculates the average distance from the uniform distributed points in the true Pareto front to their nearest solutions. Let P^* is a set of points uniformly sampled along the true Pareto front, $\mathbf{S}$ is the set of solutions obtained by an EMO algorithm,

$$\text{IGD}(P^*, \mathbf{S}) = \frac{\sum_{\mathbf{x} \in P^*} dist(\mathbf{x}, \mathbf{S})}{|\mathbf{S}|}, \tag{4.27}$$

where $dist(\mathbf{x}, P^*)$ is the Euclidean distance between a point P^* and its nearest solution in $\mathbf{x} \in S$. Note that the calculation of IGD requires the prior knowledge of the PF.

The hypervolume (HV) indicator calculates the volume of the objective space enclosed by the solutions and a reference point $\mathbf{z}$.

$$\text{HV}(S) = \text{VOL}(\bigcup_{\mathbf{x} \in S} [f_1(\mathbf{x}), z_1^w] \times \ldots [f_m(\mathbf{x}), z_m^w]) \tag{4.28}$$

where VOL$(\cdot)$ indicates the Lebesgue measure. The solutions, dominated by the worst point, are discarded for HV calculation.

4.3.3 Algorithms for Comparisons

In our empirical studies, we choose five state-of-the-art constrained evolutionary multi-objective optimization (EMO) algorithms for peer comparison. We briefly describe their basic ideas as follows:

- C-MOEA/D [11]: It is a modification of the MOEA/D-DE developed by Li et al. [43], where the feasibility information is incorporated in the update procedure: 1) if the offspring is feasible while the parent is infeasible, the offspring will replace this parent; otherwise, the parent survives; 2) if both offspring and parent are infeasible, the one having the smaller CV survives; 3) if both of them are feasible, we compare the aggregation function value as usual.

- C-NSGA-III [11]: It is a modification of NSGA-III [48] where the Pareto dominance relation is replaced with the constrained dominance relation.

- C-MOEA/DD [75]: It is recently developed by Li et al. that incorporates the Pareto dominance and decomposition into a single paradigm. Similar to NSGA-II, we first divide the population into several non-domination levels, and solutions in the first several levels have a higher priority to survive to the next generation. Afterwards, the exceeded solutions are trimmed according to the density information of their associated subregion and their aggregation function values. To improve the population diversity, an infeasible solution can survive to the next generation in case it is in an isolated subregion.

- I-DBEA [74]: It is a decomposition-based evolutionary algorithm where the balance between convergence and diversity is maintained by two independent distance measures and a simple preemptive distance comparison scheme for solution association. In addition, I-DBEA uses an adaptive epsilon formulation to deal with constraints.

- CMOEA [77]: It uses an adaptive penalty function and a distance measure to handle constraints. In particular, these two functions are developed upon the objective

function values and the sum of CVs of each solution. By doing so, the original constrained multi-objective optimization problem (CMOP) is transformed to a unconstrained format. In particular, the corresponding objective functions take account of both the optimality and feasibility. Afterwards, the classic NSGA-II is used to optimize this newly formed unconstrained MOP.

All algorithms use the simulated binary crossover [129] and the polynomial mutation [130] for offspring generation. The corresponding parameters are set in the Table 4.2.

Table 4.2: Parameter settings for reproduction operators

Parameter	Setting
crossover probability p_c	0.9
mutation probability p_m	$\frac{1}{n}$
crossover index μ_c	30
mutation index μ_m	20
number of runs	21

4.3.4 Results on C-DTLZ Benchmark Suite

The comparison results of IGD values and HV values are given in Table 4.3 and Table 4.4. The results were obtained from 21 runs. Generally speaking, our proposed C-TAEA produces superior IGD values on most test instances.

Although the feasible region of C1-DTLZ1 is only a narrow region above the PF, it actually does not pose any difficulty to all algorithms. C3-DTLZ1 and C3-DTLZ4, the original PF of the baseline problem becomes infeasible when considering the constraints while the new PF is formed by the feasible boundaries. In terms of the constraint handling, this type of problems does not provide too much difficulty. As a result, the C-TAEA cannot achieve better results than other algorithms .

For C1-DTLZ3, C-TAEA shows the best performance on all 3-to 15-objective problem instances. As the selection strategy for DA does not take the feasibility information into account, it can push the solutions of the DA toward the PF without considering the infeasible barrier. Once the DA passing though the barrier, CA can converge towards the

Table 4.3: Comparison results on IGD metric (median and IQR) for C-TAEA and the other peer algorithms on C-DTLZ benchmark suite

	m	C-TAEA	C-NSGA-III	C-MOEA/D	C-MOEA/DD	I-DBEA	CMOEA
C1-DTLZ1	3	2.069E-2(1.33E-5)	**2.037E-2(7.06E-5)**‡	2.110E-2(3.17E-4)†	2.116E-2(4.75E-4)†	2.180E-2(6.03E-6)†	2.140E-2(5.45E-4)†
	5	**5.278E-2(1.30E-3)**	5.427E-2(1.62E-3)†	5.294E-2(7.79E-5)†	5.287E-2(1.81E-5)	5.285E-2(6.62E-5)	5.284E-2(1.97E-5)
	8	**9.912E-2(1.60E-3)**	1.009E-1(1.59E-3)	1.006E-1(6.93E-4)	1.024E-1(1.86E-3)†	1.009E-1(5.30E-4)	1.008E-1(5.76E-4)
	10	1.061E-1(3.82E-3)	**1.038E-1(8.86E-3)**‡	1.074E-1(7.81E-2)	1.065e-1(9.08E-2)	1.072E-1(7.87E-3)	1.072E-1(3.39E-3)
	15	**2.233E-1(8.02E-4)**	2.351E-1(3.40E-3)†	2.608E-1(7.62E-3)†	2.490E-1(6.53E-3)†	2.506E-1(4.47E-3)†	2.611E-1(7.25E-3)†
C1-DTLZ3	3	**5.661E-2(8.49E-3)**	8.020E+0(4.22E-3)†	8.007E+0(1.72E-3)†	8.012E+0(1.08E-3)†	8.013E+0(7.59E-3)†	8.007E+0(2.07E-3)†
	5	**5.364E-1(9.03E-1)**	1.302E+1(3.96E-2)†	1.304E+1(4.41E-3)†	1.305E+1(1.12E+1)†	1.303E+1(4.79E-3)†	1.304E+1(9.23E-3)†
	8	**4.115E-1(1.31E-2)**	1.180E+1(8.59E-2)†	1.300E+1(2.64E-3)†	1.301E+1(4.47E-4)†	1.300E+1(6.98E-3)†	1.309E+1(1.84E-2)†
	10	**3.896E-1(8.75E-2)**	1.430E+1(3.30E-2)†	1.414E+1(1.93E-2)†	1.414E+1(7.36E-3)†	1.416E+1(6.11E-3)†	1.412E+1(2.90E-2)†
	15	**8.749E-1(3.30E-2)**	1.470E+1(5.33E-3)†	1.466E+1(8.22E-2)†	1.461E+1(4.30E-2)†	1.463E+1(1.26E-2)†	1.461E+1(6.31E-2)†
C2-DTLZ2	3	**1.594E-2(2.95E-3)**	9.043E-1(1.25E-4)†	9.069E-1(3.74E-1)†	5.648E-1(3.67E-1)†	9.069E-1(1.76E-3)†	9.069E-1(1.05E-2)†
	5	**3.386E-1(1.46E-1)**	1.068E+0(2.59E-5)†	4.863E-1(5.93E-1)†	1.069E+0(3.97E-2)†	1.070E+0(1.54E-3)†	1.074E+0(4.35E-3)†
	8	**1.310E-4(8.22E-4)**	1.206E+0(1.25E-5)†	1.220E+0(7.64E-3)†	1.237E+0(2.27E-6)†	1.051E+0(1.84E-1)†	1.223E+0(6.64E-4)†
	10	**2.600E-5(1.03E-6)**	1.241E+0(7.00E-6)†	1.254E+0(5.57E-3)†	1.273E+0(1.28E-5)†	1.263E+0(1.46E-1)†	1.257E+0(4.48E-3)†
	15	**5.658E-1(2.38E-3)**	1.287E+0(3.34E-4)†	1.317E+0(6.43E-2)†	1.320E+0(7.21E-1)†	1.315E+0(3.64E-2)†	1.316E+0(3.79E-2)†
C3-DTLZ1	3	**4.311E-2(1.22E-4)**	7.653E-2(1.40E-3)†	4.344E-2(2.86E-2)†	9.344E-2(1.98E-4)†	4.435E-2(4.79E-3)†	4.435E-2(1.22E-3)†
	5	1.073E-1(3.06E-5)	1.124E-1(2.76E-3)†	**1.073E-1(5.84E-5)**	1.438E-1(5.19E-4)†	1.074E-1(6.95E-6)	1.077E-1(3.30E-4)
	8	**1.993E-1(8.34E-3)**	2.052E-1(4.98E-3)	2.009E-1(4.97E-3)	2.460E-1(1.11E-4)†	2.031E-1(2.07E-3)	2.011E-1(8.72E-4)
	10	**2.104E-1(2.27E-4)**	2.310E-1(2.52E-2)†	2.301E-1(2.72E-3)†	2.655E-1(7.30E-3)†	2.304E-1(5.21E-3)†	2.303E-1(3.30E-3)†
	15	**3.463E-1(4.76E-3)**	3.686E-1(1.41E-2)†	3.989E-1(8.25E-3)†	3.688E-1(2.49E-2)†	3.680E-1(8.14E-2)†	3.909E-1(5.29E-2)†
C3-DTLZ4	3	**4.789E-1(2.00E-6)**	4.838E-1(1.03E-4)†	4.841E-1(4.21E-3)†	4.848E-1(2.57E-4)†	4.824E-1(3.57E-4)†	4.813E-1(8.11E-4)†
	5	**4.170E-1(5.51E-4)**	4.358E-1(5.65E-3)†	4.484E-1(4.89E-3)†	4.249E-1(5.17E-3)†	4.430E-1(5.07E-3)†	4.389E-1(1.36E-2)†
	8	5.049E-1(4.77E-4)	**5.020E-1(5.33E-4)**	5.268E-1(7.46E-3)†	6.481E-1(1.35E-4)†	5.234E-1(6.96E-3)†	5.236E-1(3.33E-4)†
	10	5.604E-1(3.19E-3)	**5.571E-1(5.34E-3)**	5.651E-1(1.18E-3)	5.735E-1(4.11E-3)†	5.643E-1(2.22E-2)	5.645E-1(8.09E-2)
	15	7.587E-1(5.23E-3)	7.627E-1(3.79E-2)†	7.589E-1(4.40E-2)†	**7.587E-1(3.78E-2)**	7.590E-1(8.28E-3)†	7.589E-1(2.25E-2)†

† denotes the performance of C-TAEA is significantly better than the other peers according to the Wilcoxon's rank sum test at a 0.05 significance level; ‡ denotes the corresponding algorithm significantly outperforms C-TAEA.

Table 4.4: Comparison results on HV metric (median and IQR) for C-TAEA and the other peer algorithms on C-DTLZ benchmark suite

	m	C-TAEA	C-NSGA-III	C-MOEA/D	C-MOEA/DD	I-DBEA	CMOEA
C1-DTLZ1	3	1.3042(1.01E-3)	1.3020(1.89E-3)†	**1.3043(5.43E-4)**	1.3043(1.07E-3)	1.3033(2.42E-4)	1.3039(1.60E-3)
	8	2.1435(3.02E-3)	2.1431(6.59E-4)	2.1436(1.00E-6)	**2.1436(8.00E-6)**	2.1436(6.01E-6)	2.1436(1.35E-6)
	10	2.5937(2.01E-6)	**2.5940(3.52E-4)**	2.5937(1.02E-6)	2.5937(5.11E-6)	2.5937(1.03E-6)	2.5937(2.03E-6)
	15	**4.1022(3.83E-2)**	4.0812(8.86E-2)	4.0072(7.77E-2)†	4.0277(9.30E-2)	4.0911(7.93E-2)	4.0288(3.35E-2)
C1-DTLZ3	3	**0.7351(4.00E-2)**	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†
	5	**0.1761(1.76E-1)**	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†
	8	**1.5943(8.54E-2)**	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†
	10	**2.5132(6.34E-1)**	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†
	15	**2.1825(3.86E-2)**	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†
C2-DTLZ2	3	**0.4130(2.81E-4)**	0.1225(1.00E-5)†	0.1225(1.11E-1)†	0.0000(1.21E-1)†	0.1225(3.50E-5)†	0.1225(6.50E-5)†
	5	**0.8607(2.50E-1)**	0.1482(3.00E-6)†	0.6117(4.63E-1)†	0.1482(7.30E-3)†	0.1482(6.13E-2)†	0.1482(5.21E-4)†
	8	**1.1426(6.57E-3)**	0.1973(2.90E-5)†	0.1973(3.12E-6)†	0.1949(7.21E-3)†	0.3764(1.79E-1)†	0.1973(2.32E-6)†
	10	**1.5937(2.70E-5)**	0.2387(1.20E-5)†	0.2387(2.03E-1)†	0.2358(7.21E-2)†	0.2386(2.17E-1)†	0.2387(3.71E-6)†
	15	**2.4033(6.90E-2)**	0.3840(5.90E-3)†	0.3843(7.63E-2)†	0.3797(9.26E-2)†	0.3845(5.40E-2)†	0.3843(4.26E-2)†
C3-DTLZ1	3	**1.3015(5.30E-4)**	1.1499(1.46E-2)	1.1253(2.37E-2)†	1.1086(5.30E-4)†	1.1310(3.56E-2)	1.1427(6.03E-6)
	5	**1.5781(2.30E-4)**	1.5736(1.28E-2)	1.5776(7.27E-4)	1.5656(2.20E-5)†	1.5780(1.04E-4)	1.5779(7.30E-5)
	8	2.1386(5.66E-4)	2.1332(3.07E-3)	**2.1386(3.14E-4)**	2.1367(4.00E-6)	2.1385(5.80E-5)	2.1386(9.00E-6)
	10	2.5929(2.20E-5)	2.5890(3.95E-3)	**2.5929(1.50E-5)**	2.5926(1.72E-2)	2.5929(7.36E-2)	2.5929(2.11E-2)
	15	4.1422(3.68E-1)	**4.1769(5.67E-1)**	4.3031(4.29E-2)	4.1701(7.82E-2)	4.1202(5.30E-2)	4.3063(8.62E-2)
C3-DTLZ4	3	8.4280(1.23E-2)	**8.4280(1.30E-3)**	8.4165(6.70E-3)†	8.4161(1.35E-2)†	8.4150(9.29E-2)†	8.4166(8.02E-2)†
	5	**49.5453(2.20E-3)**	49.5451(7.20E-2)	49.5330(5.80E-3)†	49.5327(6.90E-3)†	49.5346(1.67E-2)†	49.5257(3.93E-2)†
	8	546.4971(4.56E-2)	**546.4971(1.10E-3)**	546.4951(4.20E-3)	546.4943(2.29E-2)	546.4933(3.73E-2)	546.4942(4.21E-2)
	10	2654.4042(9.19E-2)	**2654.4042(7.84E-2)**	2654.4042(5.37E-2)	2654.4041(1.98E-2)	2654.4042(3.24E-2)	2654.4042(3.30E-2)
	15	**136803.0202(4.13E-2)**	136802.2201(3.70E-2)†	136802.2302(5.26E-2)†	136802.1233(9.10E-2)†	136802.1921(9.80E-0)†	136802.0201(6.23E-1)†

† denotes the performance of C-TAEA is significantly better than the other peers according to the Wilcoxon's rank sum test at a 0.05 significance level; ‡ denotes the corresponding algorithm significantly outperforms C-TAEA.

PF with the help of DA. This feature makes C-TAEA is the only algorithm that can cross the infeasible barrier, as shown in Figure 4.9.

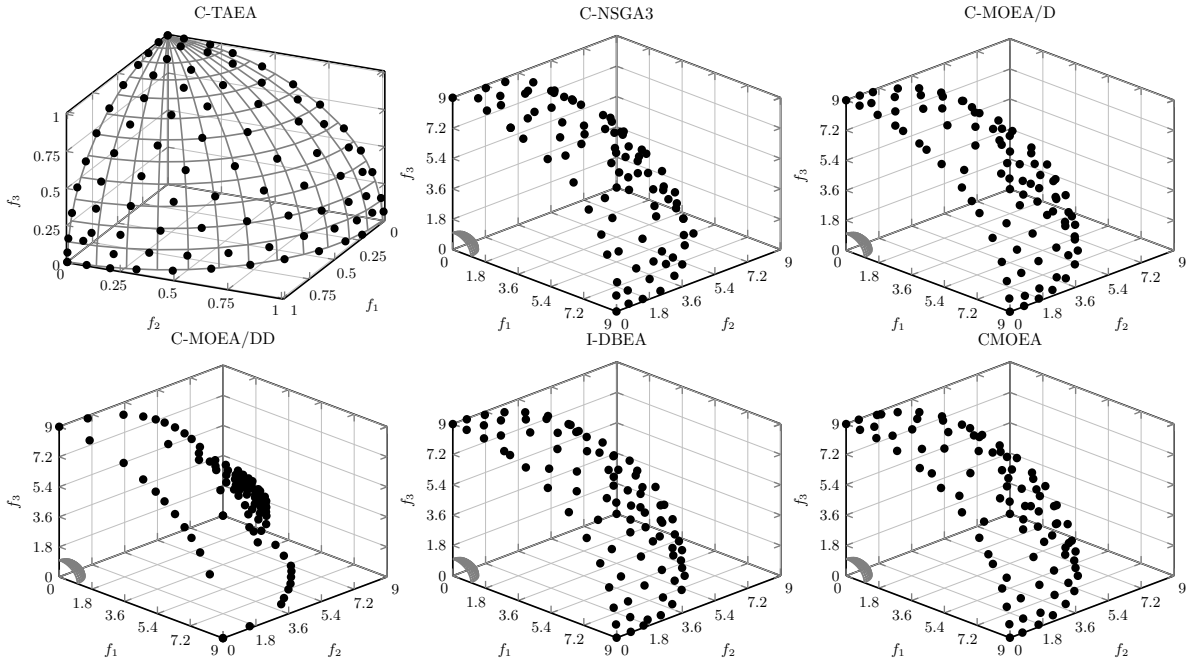


Figure 4.9: Scatter plots of the population obtained by C-TAEA and the peer algorithms on C1-DTLZ3 (median IGD value).

The feasible region of the C2-problems, i.e. C2-DTLZ2, is disjointedly distributed along the PF. All algorithms do not have any difficulty in finding at least one feasible PF segment, whereas only C-TAEA can find all disparately distributed small feasible PF segments as shown in Figure 4.10. The reason that leads to this phenomenon is similar to C1-DTLZ3. Specifically, each feasible segment is small when setting a small r in C2-DTLZ2, thus different feasible segments are separated by large infeasible barriers. In this case, if an algorithm finds one of the feasible PF segments, it hardly has any sufficient selection pressure to jump over this locally feasible PF segment. However, due to the existence of the DA in C-TAEA, it complements the coverage of the CA. As shown in Figure 4.11, solutions in the CA and the DA perfectly complements each other in terms of the coverage over the PF. Thus the DA helps the CA to explore new feasible segments.

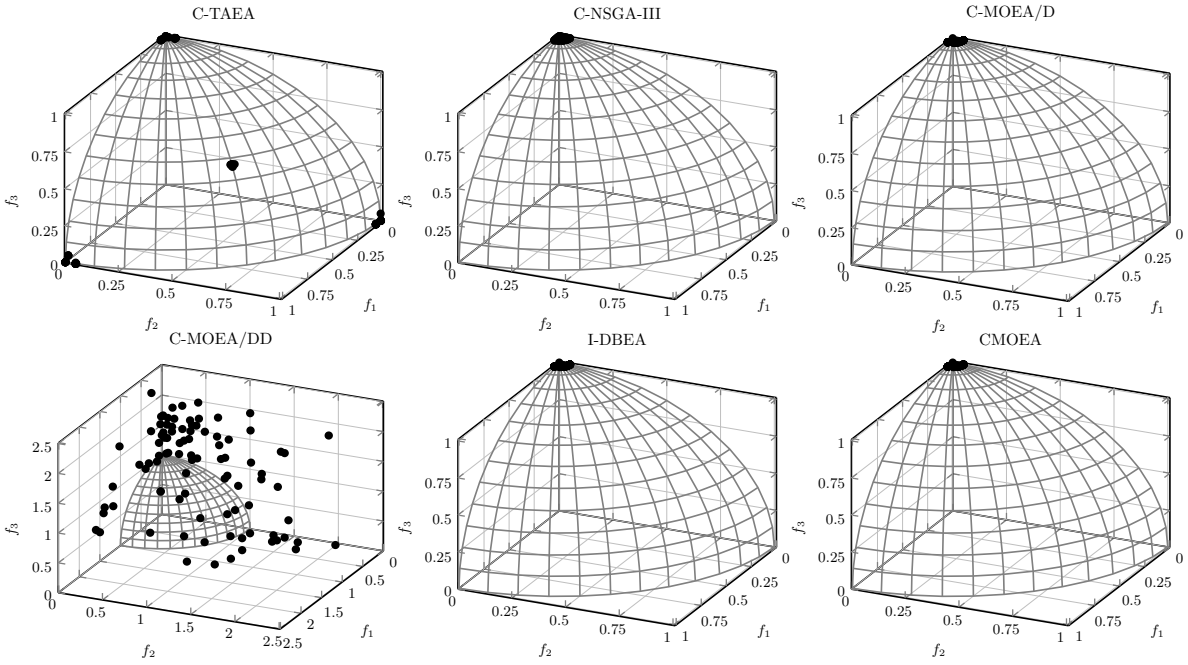


Figure 4.10: Scatter plots of the population obtained by C-TAEA and the peer algorithms on C2-DTLZ2 (median IGD value).

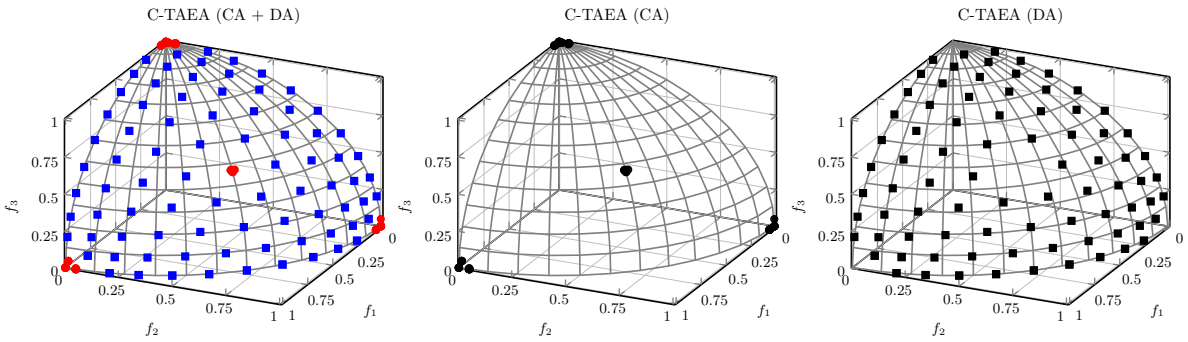


Figure 4.11: Comparison of the solutions finally obtained in CA and DA on C2-DTLZ2 (median IGD value).

4.3.5 Results on DC-DTLZ Benchmark Suite

The comparison results of IGD and HV values on the DC-DTLZ benchmark suite are given in Table 4.6 and Table 4.7 respectively. The results were obtained from 21 runs. From these results, it is obvious to see the overwhelmingly superior performance of C-TAEA over the other peer algorithms, given the observation that C-TAEA obtains the best IGD and HV values in all comparisons. The following paragraphs try to decipher the potential reasons that lead to the ineffectiveness of the other peer algorithms.

Let us start from the DC1 problem. The constraints restrict the feasible region to a couple of narrow cone-shaped strips. Similar to C2-DTLZ2, the other peer algorithms have a risk of being trapped in one feasible region thus fail to find all feasible PF segments. However, DC1-DTLZ1 and DC1-DTLZ3 seem to be less challenging than C2-DTLZ2 with a small r setting, given the observation that some peer algorithms are able to find a good number of solutions in different feasible PF segments as shown in Figure 4.12 and Figure 4.13. This might be attributed to the $g(\mathbf{x})$ function of the baseline test problems, i.e., DTLZ1 and DTLZ3, which can make the crossover and mutation generate offspring far apart from their parents. Therefore, we can expect that solutions have some opportunities to jump over the locally feasible region. Nevertheless, as shown in Table 4.6 and Table 4.7, the IGD and HV values obtained by our proposed C-TAEA constantly outperform the other peer algorithms and the better results are with a statistical significance. In contrast, due to the collaboration between two complementary populations (i.e., CA explores feasible region while DA explores infeasible region), C-TAEA is able to explore the whole search space thus has the ability to jump over the local feasible region and approximate all feasible PF segments as shown in Figure 4.12 and Figure 4.13.

The DC2-problem seems to be similar to C1-DTLZ1, where the constraints make the feasible region be reduced to a thin ribbon zone above the PF. However, it is more challenging due to the fluctuation in the CV of an infeasible solution when it approaches the PF. Table 4.5 shows the number of runs, out of 51 runs in total, where feasible solutions were found. From this table, we clearly see that all algorithms, except C-TAEA,

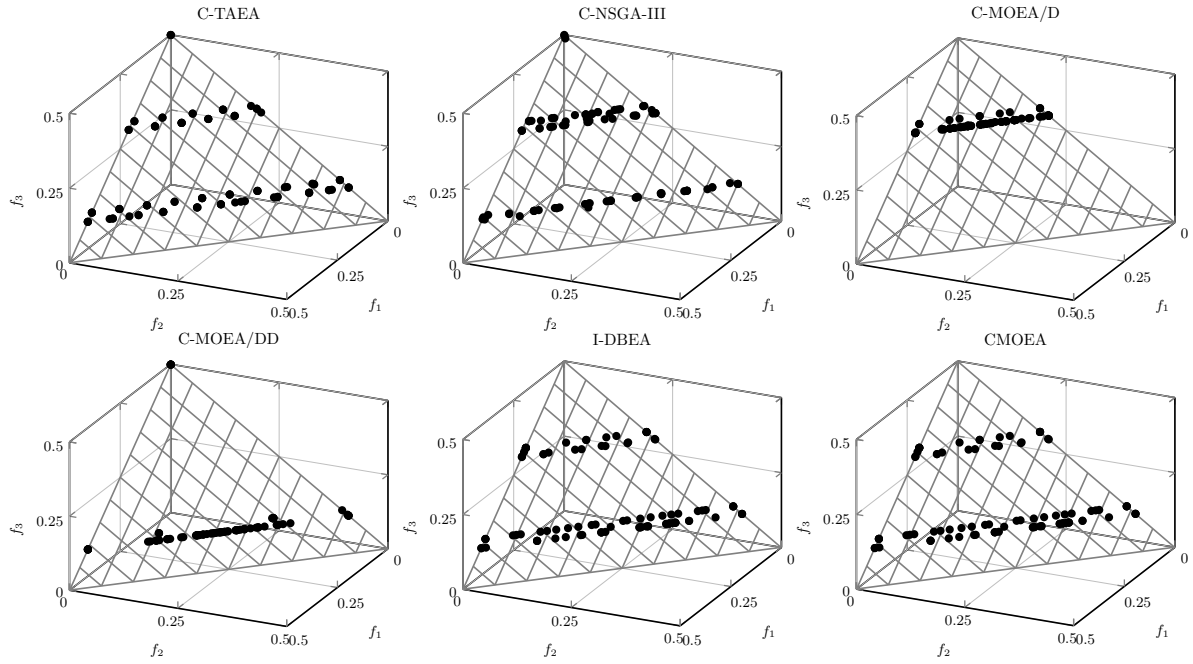


Figure 4.12: Scatter plots of the population obtained by C-TAEA and the peer algorithms on DC1-DTLZ1 (median IGD value).

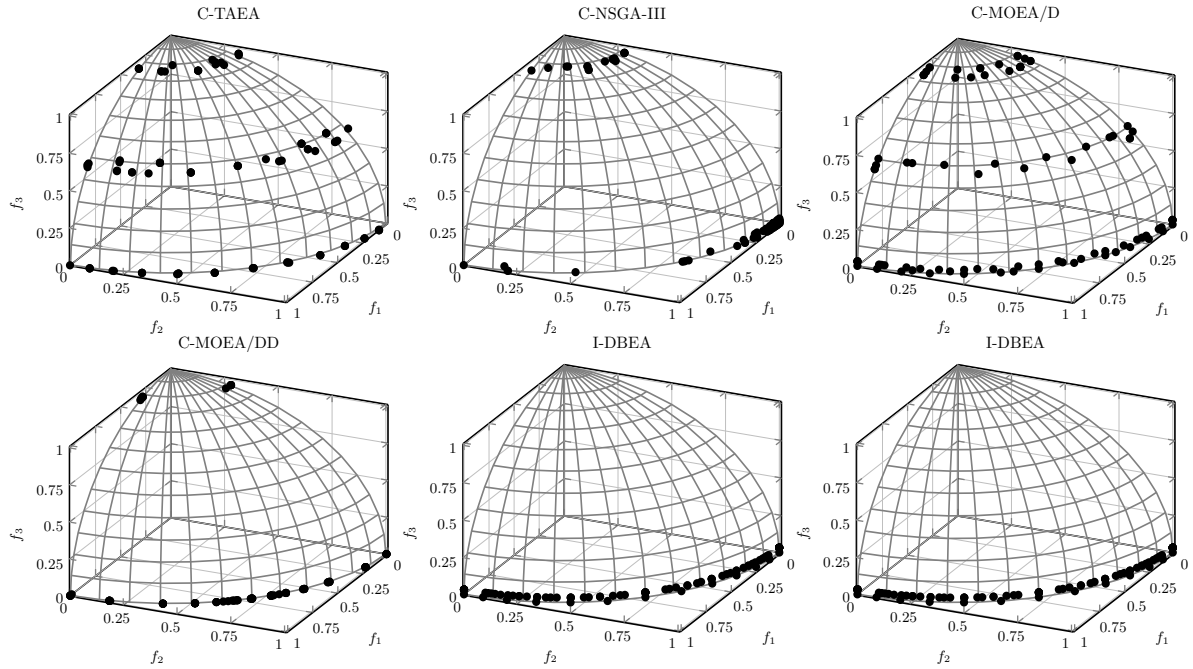


Figure 4.13: Scatter plots of the population obtained by C-TAEA and the peer algorithms on DC1-DTLZ3 (median IGD value).

can hardly find feasible solutions in most cases. This is also demonstrated from Figure 4.14 and Figure 4.15, where we can clearly see that all other peer algorithms are trapped in a region far away from the PF. According to the problem definitions of DC2-DTLZ1 and DC2-DTLZ3, all solutions obtained by the other peer algorithms are infeasible. Their failures on this type of constrained problems can be attributed to their feasibility-driven selection mechanisms, which drive the population to fluctuate between the CV's local optima. As for our proposed C-TAEA, its success can be owed to the use of the DA. In particular, the selection mechanism of the DA does not take the CV into account so that it has sufficient selection pressure to move toward the PF. As shown in Figure 4.14 and Figure 4.15, only C-TAEA finally find solutions on the PF. However, from Table 4.5, we also find that C-TAEA can end up with infeasible solutions while the other algorithms have a chance to obtain feasible solutions. This is because the crossover and mutation can generate some significantly different offspring when working on the $g(\mathbf{x})$ function of DC2-DTLZ1 and DC2-DTLZ3. And such offspring solutions have a chance to bring infeasible solutions to the feasible region.

Table 4.5: Number of runs when finding feasible solutions.

	m	C-TAEA	C-NSGA-III	C-MOEA/D	C-MOEA/DD	I-DBEA	CMOEA
DC2-DTLZ1	3	**46**	4	3	2	2	0
	5	**43**	5	1	2	1	2
	8	**33**	1	0	2	0	2
	10	**39**	0	0	1	1	0
	15	**31**	1	1	1	2	0
DC2-DTLZ3	3	**51**	5	2	1	1	2
	5	**51**	6	3	2	2	1
	8	**29**	2	0	1	2	2
	10	**37**	1	2	0	1	2
	15	**35**	2	1	0	2	2

As for the DC3 problem, its constraints are a combination of the previous two. In particular, the feasible region is restricted to a couple of segmented cone stripes. In addition, there exists the same fluctuation, as the DC2 problem, in the CV of an infeasible solution when it approaches the PF. In this case, the other peer algorithms are not only struggling on jumping over a particular locally feasible segment, but also have a significant trouble with the fluctuation (back and forth) of the population. Again, the success of our proposed C-TAEA is also attributed to the collaborative and complementary effects of

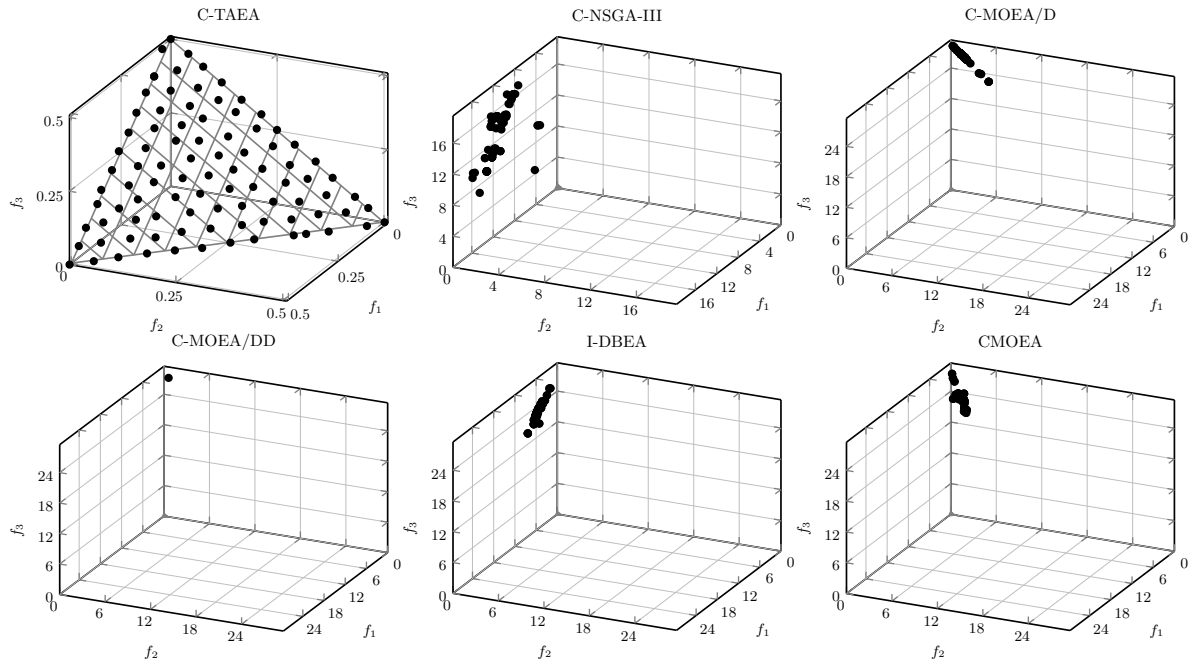


Figure 4.14: Scatter plots of the population obtained by C-TAEA and the peer algorithms on DC2-DTLZ1 (median IGD value).

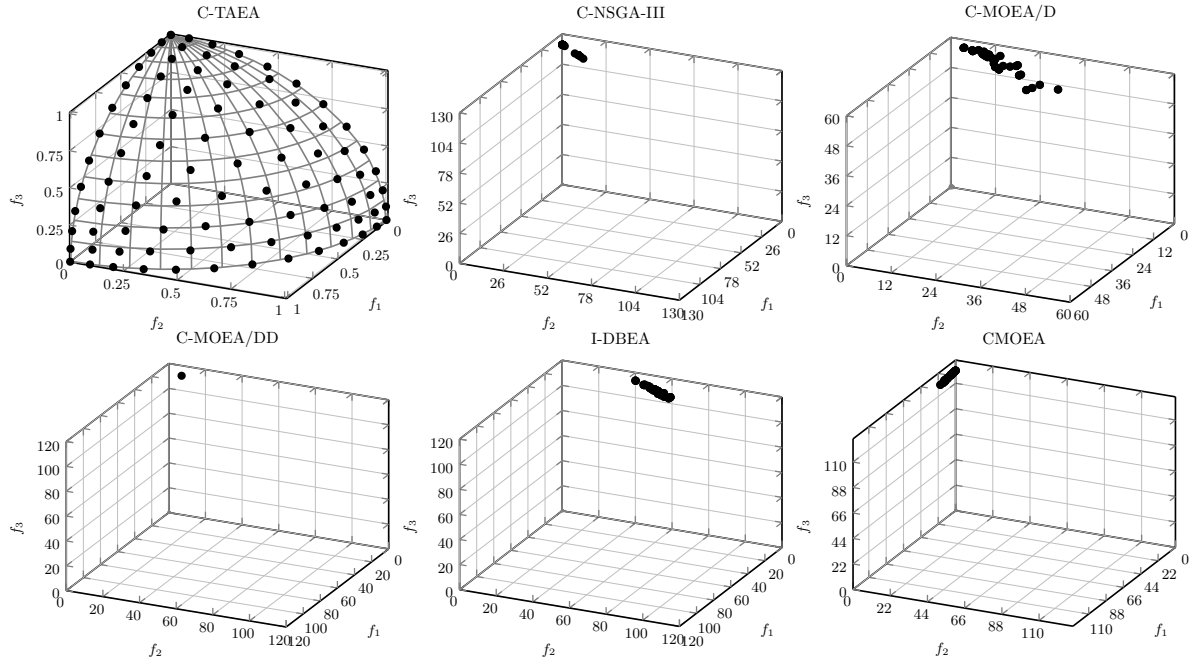


Figure 4.15: Scatter plots of the population obtained by C-TAEA and the peer algorithms on DC2-DTLZ3 (median IGD value).

two archives. As shown in Figure 4.16 and Figure 4.17, only C-TAEA finds all feasible PF segments while the other peer algorithms are stuck at some locally feasible segments away from the PF.

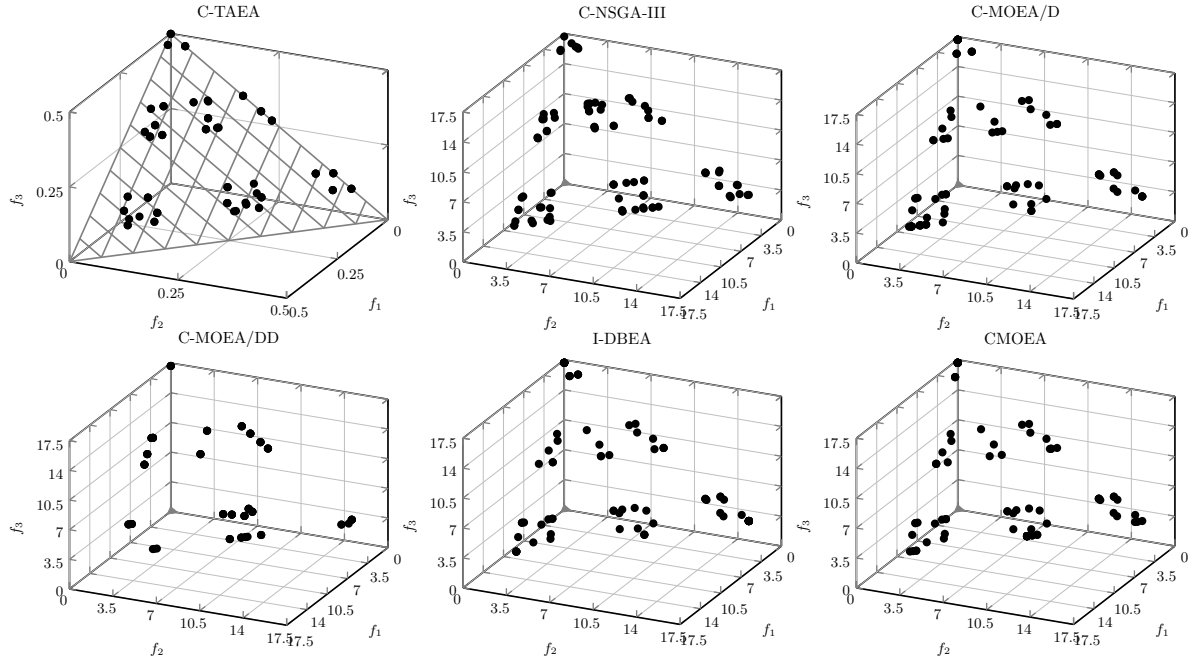


Figure 4.16: Scatter plots of the population obtained by C-TAEA and the peer algorithms on DC3-DTLZ1 (median IGD value).

4.3.6 Results on Dynamic C-DLTZ Benchmark Suite

For dynamic problems we use the MIGD mentioned in Section 3.3.2 to measure the performance of the algorithms. The comparison results of MIGD values are given in Table 4.8. The results were obtained from 21 runs. Our proposed C-TAEA produces superior IGD values on most test instances. We only calculate IGD because the HV is too slow to calculate for problems with 15 objectives. It costs about 3 days calculating the HV for a single data set. However, dynamic problems request calculating every population during the optimization process, which will cost too much time(over one year).

It should be pointed out that no experiments are done on the dynamic DC-DTLZ test benchmark suite in this thesis. The reason is that the static version DC-DTLZ series are

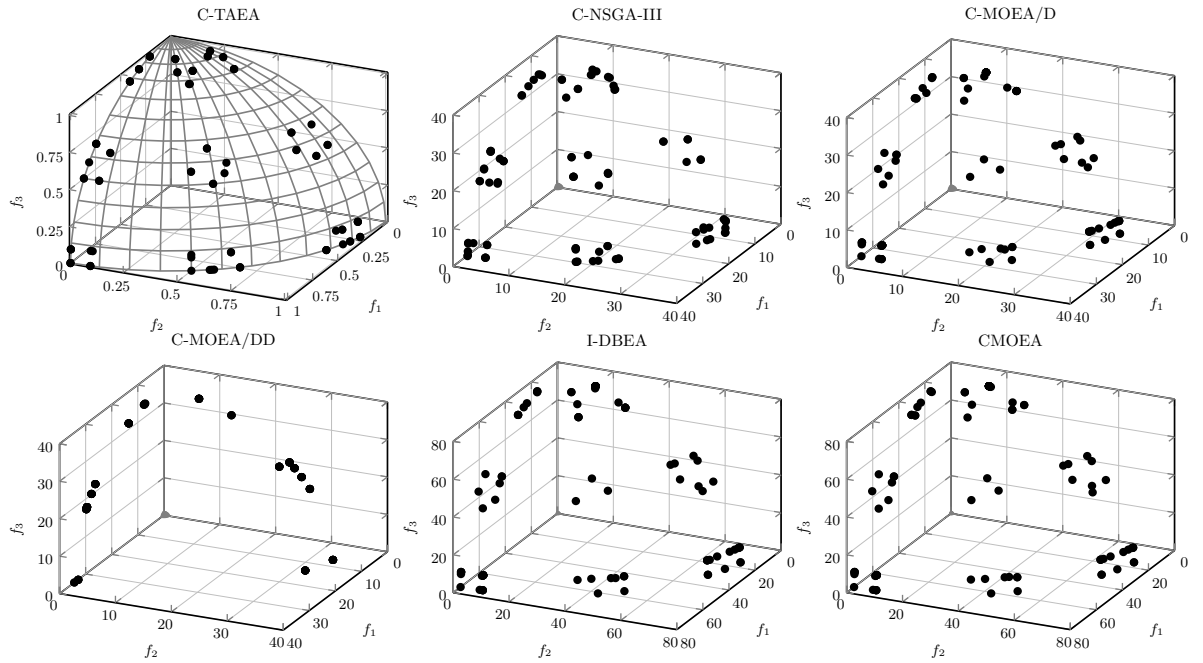


Figure 4.17: Scatter plots of the population obtained by C-TAEA and the peer algorithms on DC3-DTLZ3 (median IGD value).

Table 4.6: Comparison results on IGD metric (median and IQR) for C-TAEA and the other peer algorithms on DC-DTLZ benchmark suite

	m	C-TAEA	C-NSGA-III	C-MOEA/D	C-MOEA/DD	I-DBEA	CMOEA
DC1-DTLZ1	3	**5.638E-2(8.10E-5)**	5.990E-2(1.59E-5)†	1.835E-1(1.26E-1)†	1.042E-1(2.03E-3)†	5.843E-2(3.38E-3)†	5.843E-2(3.65E-3)†
	5	**7.301E-2(3.76E-3)**	7.655E-2(1.41E-2)†	7.327E-2(1.71E-4)†	8.705E-2(3.75E-3)†	7.344E-2(3.91E-4)†	7.640E-2(3.38E-3)†
	8	**1.086E-1(6.44E-4)**	1.104E-1(9.78E-4)†	1.414E-1(1.51E-2)†	1.175E-1(7.30E-2)†	1.290E-1(8.13E-2)†	1.291E-1(5.20E-3)†
	10	**1.189E-1(2.84E-3)**	1.206E-1(3.34E-3)†	1.524E-1(5.84E-3)†	1.278E-1(4.24E-2)†	1.545E-1(6.27E-2)†	1.529E-1(8.30E-3)†
	15	**1.753E-1(1.83E-2)**	1.984E-1(4.11E-3)†	2.017E-1(6.17E-2)†	1.772E-1(5.25E-3)†	2.070E-1(7.79E-2)†	1.986E-1(2.06E-2)†
DC1-DTLZ3	3	**1.466E-1(7.62E-4)**	2.720E-1(1.31E-1)†	1.349E-1(3.77E-1)†	2.908E-1(1.18E-1)†	5.140E-1(3.75E-1)†	5.140E-1(3.77E-1)†
	5	**2.083E-1(2.54E-3)**	2.040E-1(1.01E-2)†	3.947E-1(1.18E-4)†	2.318E-1(7.30E-4)†	3.948E-1(8.69E-4)†	3.947E-1(2.47E-4)†
	8	**3.405E-1(8.35E-5)**	4.062E-1(3.03E-2)†	4.330E-1(1.68E-3)†	3.639E-1(5.28E-2)†	4.344E-1(8.13E-3)†	3.422E-1(4.01E-2)†
	10	**3.886E-1(3.18E-3)**	4.586E-1(4.89E-2)†	4.596E-1(4.06E-3)†	4.304E-1(9.14E-3)†	4.456E-1(2.23E-3)†	4.235E-1(5.23E-3)†
	15	**8.009E-1(5.10E-3)**	8.287E-1(6.23E-3)†	8.456E-1(6.28E-2)†	8.034E-1(5.80E-3)†	8.300E-1(1.26E-2)†	8.144E-1(7.20E-3)†
DC2-DTLZ1	3	**2.199E-2(8.44E-3)**	\	\	\	\	\
	5	**5.371E-2(3.07E-2)**	\	\	\	\	\
	8	**9.937E-2(—)**	\	\	\	\	\
	10	**1.048E-1(8.65E-3)**	\	\	\	\	\
	15	**2.308E-1(—)**	\	\	\	\	\
DC2-DTLZ3	3	**5.498E-2(6.78E-2)**	\	\	\	\	\
	5	**1.667E-1(9.36E-3)**	\	\	\	\	\
	8	**5.674E+1(—)**	\	\	\	\	\
	10	**3.836E-1(—)**	\	\	\	\	\
	15	**7.959E-1(—)**	\	\	\	\	\
DC3-DTLZ1	3	**5.034e-2(1.72E-4)**	9.745E+0(5.64E-3)†	9.746E+0(7.80E-3)†	9.789E+0(8.76E-4)†	9.745E+0(2.02E-3)†	9.755E+0(1.29E-2)†
	5	**8.554E-1(1.29E-3)**	7.702E+0(2.60E-2)†	8.305E+0(1.78E-1)†	8.467E+0(1.21E-1)†	1.847E+1(1.03E+1)†	8.408E+0(1.71E-3)†
	8	**1.250E-1(6.01E-1)**	6.450E+0(2.30E+0)†	9.729E+0(2.03E+0)†	6.988E+0(3.74E-3)†	8.409E+0(1.30E-2)†	5.938E+0(2.83E+0)†
	10	**2.332E-1(5.29E-3)**	5.598E+0(8.71E-2)†	2.120E+1(7.29E-3)†	6.004E+0(8.26E-3)†	8.432E+0(5./9E-2)†	7.306E+0(1.93E-3)†
	15	**1.837E-1(3.43E-5)**	5.431E+0(4.38E-1)†	2.567E+1(1.10E+1)†	2.346E-1(7.51E+0)†	7.204E+0(1.76E+1)†	2.584E+1(1.66E+1)†
DC3-DTLZ3	3	**1.250E-1(8.04E-4)**	3.334E+1(7.20E-2)†	3.335E+1(6.20E-2)†	3.337E+1(2.54E-2)†	7.335E+1(8.46E-2)†	7.335E+1(4.52E-2)†
	5	**2.219E-1(9.30E-3)**	3.349E+1(5.57E-3)†	3.340E+1(3.75E-3)†	3.341E+1(4.86E-4)†	3.340E+1(7.59E-1)†	3.339E+1(2.28E-2)†
	8	**3.429E-1(8.37E-2)**	3.360E+1(3.52E-3)†	3.350E+1(1.88E-2)†	3.343E+1(5.02E-3)†	3.369E+1(3.39E-3)†	3.359E+1(7.59E-3)†
	10	**3.835E-1(1.30E-3)**	3.362E+1(9.10E-2)†	7.377E+1(9.92E-3)†	7.346E+1(8.57E-3)†	7.376E+1(7.36E-2)†	7.377E+1(4.91E-2)†
	15	**7.872E-1(2.33E-2)**	7.411E+1(3.62E-3)†	1.541E+2(8.61E-3)†	7.407E+1(9.35E-2)†	7.416E+1(4.29E-2)†	7.407E+1(5.49E-2)†

† denotes the performance of C-TAEA is significantly better than the other peers according to the Wilcoxon's rank sum test at a 0.05 significance level; ‡ denotes the corresponding algorithm significantly outperforms C-TAEA. \ denotes the median metric value is not available, while — denotes the IQR is not available.

Table 4.7: Comparison results on HV metric (median and IQR) for C-TAEA and the other peer algorithms on DC-DTLZ benchmark suite

	m	C-TAEA	C-NSGA-III	C-MOEA/D	C-MOEA/DD	I-DBEA	CMOEA
DC1-DTLZ1	3	**1.2006(1.70E-2)**	1.1982(9.96E-2)†	0.9631(2.81E-2)†	1.1845(3.05E-2)†	1.1883(9.25E-2)†	1.1883(8.40E-3)†
	5	**1.4783(3.27E-2)**	1.4725(3.36E-2)†	1.4783(5.05E-2)†	1.4762(8.46E-2)†	1.4782(7.29E-2)†	1.4779(3.13E-2)†
	8	**1.9682(1.24E-2)**	1.9347(8.57E-2)†	1.9660(8.95E-2)†	1.9655(5.82E-2)†	1.9678(5.45E-2)†	1.9676(8.14E-2)†
	10	**2.3890(3.67E-3)**	2.3113(4.34E-2)†	2.3792(3.64E-2)†	2.3801(4.30E-3)†	2.3797(7.50E-2)†	2.3810(8.70E-3)†
	15	**4.1012(1.73E-2)**	4.0031(2.76E-2)†	4.0122(1.78E-2)†	4.0823(7.88E-2)†	4.0911(5.84E-2)†	4.0532(4.03E-2)†
DC1-DTLZ3	3	**0.6339(6.51E-2)**	0.5088(7.54E-2)†	0.6694(3.99E-2)†	0.5386(9.32E-2)†	0.4545(6.00E-3)†	0.4545(6.76E-2)†
	5	**1.2656(3.68E-2)**	1.2582(7.39E-2)†	1.1463(1.20E-3)†	1.1712(2.26E-2)†	1.1467(5.86E-2)†	1.1462(9.40E-3)†
	8	**1.9829(5.39E-2)**	1.8461(7.95E-2)†	1.9301(5.70E-2)†	1.9640(4.34E-2)†	1.9284(3.78E-2)†	1.9793(4.67E-2)†
	10	**2.5181(6.01E-2)**	2.3880(9.70E-3)†	2.4903(9.02E-2)†	2.5083(3.17E-2)†	2.4902(4.92E-2)†	2.4986(6.52E-2)†
	15	**4.1700(7.56E-2)**	4.0321(3.01E-2)†	4.0422(2.80E-2)†	4.0282(2.86E-2)†	3.9123(4.41E-2)†	4.1028(8.65E-2)†
DC2-DTLZ1	3	**1.3010(5.30E-4)**	\	\	\	\	\
	5	**1.5781(2.30E-4)**	\	\	\	\	\
	8	**2.1386(—)**	\	\	\	\	\
	10	**2.5929(2.20E-5)**	\	\	\	\	\
	15	**4.1422(—)**	\	\	\	\	\
DC2-DTLZ3	3	**0.7377(3.68E-2)**	\	\	\	\	\
	5	**1.3087(7.23E-2)**	\	\	\	\	\
	8	**2.0013(—)**	\	\	\	\	\
	10	**2.5101(—)**	\	\	\	\	\
	15	**4.0832(—)**	\	\	\	\	\
DC3-DTLZ1	3	**1.2134(1.10E-5)**	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†
	5	**1.4751(2.53E-3)**	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†
	8	**1.9429(1.94E+0)**	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†
	10	**2.3933(1.05E-1)**	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†
	15	**4.0012(4.32E-2)**	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†
DC3-DTLZ3	3	**0.6298(4.74E-2)**	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†
	5	**1.1880(2.35E-2)**	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†
	8	**1.7614(9.28E-2)**	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†
	10	**2.3748(6.13E-2)**	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†
	15	**4.1326(1.28E-2)**	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†	0.0000(0.00E+0)†

† denotes the performance of C-TAEA is significantly better than the other peers according to the Wilcoxon's rank sum test at a 0.05 significance level; ‡ denotes the corresponding algorithm significantly outperforms C-TAEA. \ denotes the median metric value is not available, while — denotes the IQR is not available.

Table 4.8: Comparison results on IGD metric (median and IQR) for C-TAEA and the other peer algorithms on Dynamic C-DTLZ benchmark suite

	m	C-TAEA	C-NSGA-III	C-MOEA/D	C-MOEA/DD	I-DBEA	CMOEA
Dynamic C1-DTLZ1	3	3	2.469E-2(1.33E-5)‡	**2.094E-2(7.71E-5)**	2.181E-2(3.63E-4)†	2.384E-2(4.98E-4)†	2.433E-2(6.74E-6)†
	5	5	**4.812E-2(1.30E-3)**	6.054E-2(1.93E-3)†	5.422E-2(9.01E-5)†	6.193E-2(2.05E-5)†	5.679E-2(6.89E-5)†
	8	8	1.006E-1(1.61E-3)	1.057E-1(1.89E-3)†	**1.028E-1(7.83E-4)**	1.177E-1(1.87E-3)†	1.136E-1(5.38E-4)†
	10	10	1.052E-1(4.22E-3)	**1.101E-1(9.48E-3)**	1.201E-1(8.49E-2)†	1.108E-1(9.14E-2)†	1.123E-1(8.28E-3)†
	15	15	**2.323E-1(8.68E-4)**	2.495E-1(3.69E-3)†	2.749E-1(9.08E-3)†	2.641E-1(6.62E-3)†	2.828E-1(5.29E-3)†
Dynamic C1-DTLZ3	3	3	**6.265E-2(8.94E-3)**	8.677E+0(4.71E-3)†	8.975E+0(1.96E-3)†	8.388E+0(1.24E-3)†	8.148E+0(8.73E-3)†
	5	5	**6.400E-1(1.03E+0)**	1.332E+1(4.11E-2)†	1.180E+1(4.78E-3)†	1.339E+1(1.14E+1)†	1.360E+1(5.07E-3)†
	8	8	**4.628E-1(1.57E-2)**	1.223E+1(8.60E-2)†	1.222E+1(2.83E-3)†	1.358E+1(5.29E-4)†	1.281E+1(7.27E-3)†
	10	10	**4.659E-1(9.08E-2)**	1.662E+1(3.78E-2)†	1.687E+1(2.22E-2)†	1.555E+1(8.57E-3)†	1.529E+1(6.98E-3)†
	15	15	**9.390E-1(3.66E-2)**	1.702E+1(5.96E-3)†	1.504E+1(9.34E-2)†	1.665E+1(4.56E-2)†	1.675E+1(1.44E-2)†
Dynamic C2-DTLZ2	3	3	**1.887E-2(3.11E-3)**	9.820E-1(1.33E-4)†	9.223E-1(3.78E-1)†	6.303E-1(3.89E-1)†	1.068E+0(1.90E-3)†
	5	5	**3.502E-1(1.54E-1)**	1.309E+0(2.84E-5)†	4.925E-1(6.40E-1)†	1.234E+0(4.08E-2)†	1.259E+0(1.61E-3)†
	8	8	**1.327E-4(8.43E-4)**	1.304E+0(1.37E-5)†	1.455E+0(8.02E-3)†	1.289E+0(2.31E-6)†	1.304E+0(1.91E-1)†
	10	10	**2.799E-5(1.08E-6)**	1.313E+0(7.36E-6)†	1.495E+0(6.33E-3)†	1.301E+0(1.36E-5)†	1.443E+0(1.49E-1)†
	15	15	**5.938E-1(2.50E-3)**	1.325E+0(3.42E-4)†	3.242E+0(8.14E-1)†	1.509E+0(4.28E-2)†	1.491E+0(3.92E-2)†
Dynamic C3-DTLZ1	3	3	**4.412E-2(1.40E-4)**	8.730E-2(1.55E-3)†	4.870E-2(3.27E-2)†	1.071E-1(2.06E-4)†	4.802E-2(4.88E-3)†
	5	5	**1.183E-1(3.38E-5)**	1.246E-1(3.30E-3)†	1.121E-1(6.28E-5)†	1.609E-1(5.32E-4)†	1.269E-1(7.14E-6)†
	8	8	2.373E-1(9.24E-3)‡	2.339E-1(5.86E-3)†	**2.188E-1(5.26E-3)**	2.818E-1(1.12E-4)†	2.359E-1(2.41E-3)†
	10	10	**2.216E-1(2.33E-4)**	2.661E-1(2.64E-2)†	2.240E-1(2.97E-3)†	2.938E-1(8.14E-3)†	2.301E-1(5.85E-3)†
	15	15	**3.612E-1(5.56E-3)**	4.206E-1(1.57E-2)†	4.014E-1(8.98E-3)†	4.041E-1(2.55E-2)†	3.723E-1(8.64E-2)†
DynamicC3-DTLZ4	3	3	5.536E-1(2.02E-6)‡	**5.077E-1(1.30E-4)**	5.268E-1(4.48E-3)†	5.774E-1(2.81E-4)†	5.623E-1(4.08E-4)†
	5	5	**4.640E-1(6.14E-4)**	4.977E-1(5.88E-3)†	4.717E-1(4.93E-3)†	4.645E-1(5.39E-3)	4.466E-1(5.92E-3)†
	8	8	**5.322E-1(5.26E-4)**	6.018E-1(5.35E-4)†	6.029E-1(7.83E-3)†	6.703E-1(1.59E-4)†	5.687E-1(8.25E-3)†
	10	10	**5.339E-1(3.59E-3)**	5.670E-1(5.83E-3)†	6.469E-1(1.22E-3)†	6.268E-1(4.87E-3)†	5.745E-1(2.51E-2)†
	15	15	7.891E-1(5.76E-3)	7.903E-1(4.52E-2)†	8.865E-1(5.30E-2)†	8.362E-1(3.89E-2)†	**7.891E-1(8.51E-3)**

† denotes the performance of C-TAEA is significantly better than the other peers according to the Wilcoxon's rank sum test at a 0.05 significance level; ‡ denotes the corresponding algorithm significantly outperforms C-TAEA. \ denotes the median metric value is not available, while — denotes the IQR is not available.

already very challenging for all the algorithms. C-TAEA cannot successfully converged for every single run. When facing the dynamic DC-DTLZ series, none of the algorithms can achieve a well-converged population steadily (although C-TAEA is the only algorithm in the comparison can achieve satisfied population in some runs).

For dynamic C1-DTLZ1, all algorithms perform well. For the dynamic C1-DTLZ3 problem, C-TAEA is the only algorithm that successfully crosses the infeasible barrier and converges to the PF. This is because the DA in C-TAEA ignores the feasibility informations. As a result, the dynamic constraints have little impact on the performance of C-TAEA. In the meantime, other algorithms perform even worse comparing to the static situation. None of the algorithms achieve a converged population.

For the dynamic C2-DTLZ2 problem, C-TAEA performs extremely good. It is because when the feasible area moves in C2-DTLZ2, solutions become infeasible in the CA after the changes are maintained in the DA, while the feasible solutions in the DA after the changes are moving into the CA. For other algorithms, take C-NSGA-III as an example,

after every changes, the solutions becoming infeasible are directly discarded, even though after several time steps, these solutions become feasible again.

For the dynamic C3-DTLZ1 and C3-DTLZ4 problems, C-TAEA does not achieve better performance than other algorithms. The reason for this phenomenon is: all the algorithms can trace and maintain the boundary of the PF. When the boundary of the PF moves towards the PF without constraints, C-TAEA is benefited by the help of DA. While the boundary is moving away from the PF without constraints, the DA makes the algorithm perform worse.

4.4 Case Study: Water Distribution Network Optimization

Having tested C-TAEA's ability in solving dynamic constrained benchmark problems, this section tends to investigate the performance of C-TAEA and the other peer algorithms on a real-world case study about optimal design of the water distribution network (WDN). In the past decade, multi-objective optimal design and rehabilitation of a WDN has attracted an increasing attention [131]. The shift from the least-cost design to a multi-objective performance-based design advances decision makers' understanding of trade-off relationship between conflicting design objectives [132].

This study uses the Anytown WDN, one of the most popular benchmark networks, as the case study. Anytown WDN has many typical features and challenges that can be found in real-world networks, e.g., pump scheduling, tank storage provision, and fire-fighting capacity provision. The network layout is shown in Figure 4.18, where it has 35 pipes, 2 storage tanks, and 3 identical pumps delivering water from the treatment plant into the system. To meet the city expansion and increasing demands, 77 decision variables are considered, including 35 variables related to the existing pipes (with options of cleaning and lining or duplication with a parallel pipe), six new pipe diameters, 12 variables for two potential tanks, and 24 variables for the number of pumps in operation

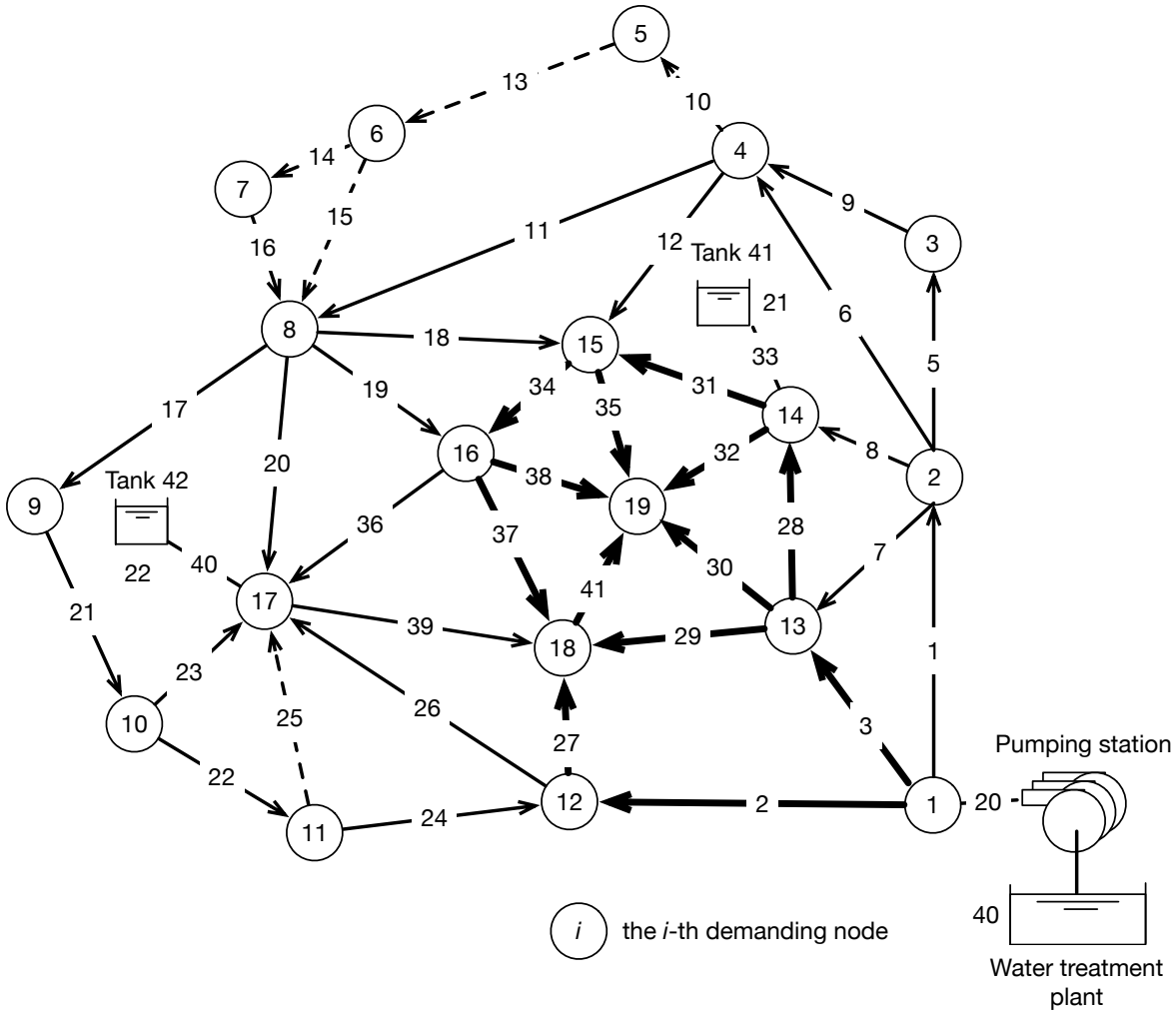


Figure 4.18: Layout of the anytown WDN.

during 24 hours of a day. In this study, the WDN design problem is formulated as a four-objective optimization problem with two constraints. In particular, we consider costs, resilience index, statistical flow entropy and water age as the objective functions.

Here briefly describes the problem formulation of the water distribution network (WDN) design, including the objective and constraint functions, used in our case study. In particular, there are four objective functions and two time-dependent constraint functions in total.

4.4.1 Objective Functions

Cost

One of the most important objective functions of the WDN design is the cost. There are many different cost metrics available in the literature. This chapter considers the following form:

$$f_{cost} = C_c + C_o, \tag{4.29}$$

where C_c is the capital cost which accounts for the costs of network components such as pipes, storage tanks; while C_o is the operating cost which measure the energy costs for pump operation. More detailed information and configurations of the costs of pipes, storage tanks and pump operation used in the WDN model, i.e., Anytown model, can be found in [133].

Resilience index

The concept of resilience index was introduced by Todini [134] to account for the reliability and availability under certain stressed conditions. Specifically, it is calculated as:

$$f_r = \frac{\sum_{j=1}^{N_n} Q_j(H_j - H_j^{min})}{(\sum_{k=1}^{N_r} Q_k \overline{H}_k + \sum_{i=1}^{N_p} P_i/\gamma) - \sum_{j=1}^{N_n} Q_j H_j}, \tag{4.30}$$

where Q_j and H_j are respectively the demand and head at j-th node, N_n is the number of nodes. $H_j^{\min}$ is the required head at the j-th node. γ is the specific weight of water, Q_k and $\overline{H}_k$ are discharge and the head of the k-th reservoir, N_r is the number of reservoirs. P_i is the power introduced into the network by the i-th pump and N_p is the number of pumps.

Statistical flow entropy

This objective function was proposed in [135], and it is formulated as:

$$f_e = S_0 + \sum_{i=1}^{N_n} F_i S_i, \tag{4.31}$$

where F_i is the fraction of the total flow that a network supplies to the i-th node; S_0 is entropy of sources or external supplies; S_i is the entropy of the i-th node. Specifically, they are calculated as:

$$\begin{aligned} S_0 &= -\textstyle\sum_{i\in I} \frac{F_{0,i}}{Q} ln(\frac{F_{0,i}}{Q}), \\ S_i &= -\frac{Q_i}{T_i} ln(\frac{Q_i}{T_i}) - \textstyle\sum_{i=1}^{N_n} \sum_{j\in N_i} \frac{F_{i,j}}{Q_i} ln(\frac{F_{ij}}{Q_i}), \end{aligned} \tag{4.32}$$

where $F_{0,i}$ is the inflow at the i-th source node, Q is the total demand for all nodes, and the set I includes all the source nodes. Q_i is the demand at the i-th node, while T_i is the total flow that reaches the i-th node. N_i is the set of nodes with pipes flow from the i-th node; $F_{i,j}$ is the volume flow rate in the pipe between the i-th and j-th nodes.

Water age

Water quality problems in a WDN often arise from interactions between water within the pipe and the pipe wall in addition to within the bulk water of storage tanks. There is a great chance for contamination and thus harm the health if the water stays in the system for a long period of time. In this case, we can see that the time required for the water to reach the customer from water sources through the network influences the water

quality. As discussed in [136], water age at a demand node is used as an indicator of water quality and is defined as the average travel time from water sources to the demand nodes. Specifically, it is calculated as:

$$f_{age} = \max_{i,t}(WA_{i,t}) \tag{4.33}$$

where $WA_{i,t}$ indicates the water age of the i-th node at time step t and it can be calculated by EPANET [137],

4.4.2 Constraints

The first constraint considered in this thesis is the satisfaction of the minimum specified pressure supplied to each node. It is formulated as:

$$H_i \geq H_i(t)^{min}, i = 1, ..., N_n \tag{4.34}$$

where H_i is the pressure at the i-th node; while $H_i(t)^{min}$ is its corresponding minimum required pressure at simulation time t; N_n is the number of demand nodes.

Moreover, the height of water in a storage tank is required to be recovered over a normal operating cycle. In other words, the volume of water in a storage tank at the end of a day must be no less than the volume at the beginning of a day. Specifically, the second constraint function is defined as:

$$V_{k,E} \geq V_{k,S}, k = 1, ..., N_t \tag{4.35}$$

where $V_{k,S}$ represents the volume of the water at a checking time of a day for the k-th tank while $V_{k,E}$ represents the corresponding volume at the other checking time of a day. N_t is the number of storage tanks in a WDN.

Note that all the above mentioned objective and constraint functions are calculated by EPANET [137]. It is a software developed by the United States Environmental Pro-

tection Agency, and is used to model the hydraulic and water quality behaviour of water distribution piping systems. In particular, EPANET is able to simulate: 1) the flow of water in each pipe; 2) the pressure at each node; 3) the height of the water in each tank; and 4) water age.

4.4.3 Results

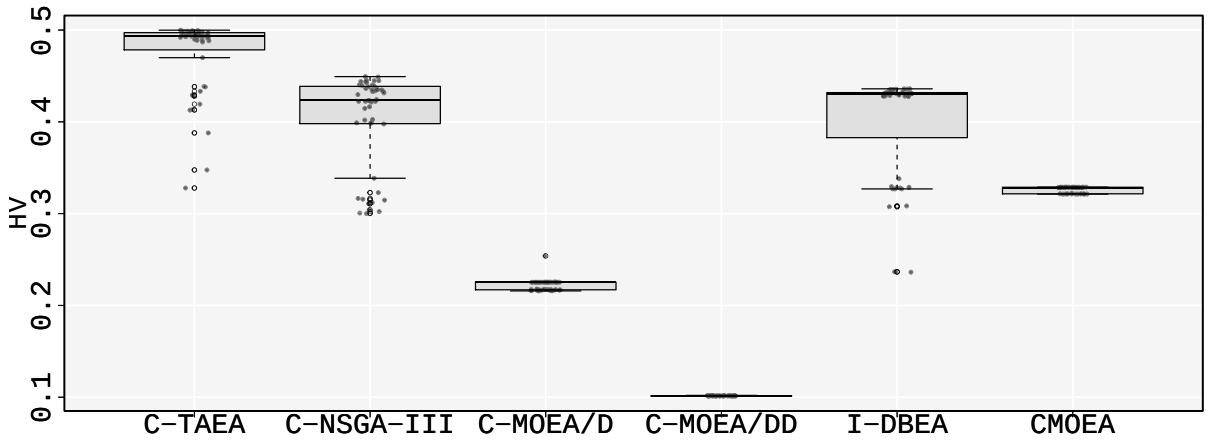


Figure 4.19: Box plots of HV obtained by different algorithms.

In the experiment, C-TAEA and the other five peer algorithms use the solution encoding scheme as suggested in [138]. The population size is set to $N = 100$, and the number of function evaluations used for each algorithm is set to $10,000 \times N$. The reproduction operators and their corresponding parameters are still set the same as before. Since the true PF is unknown for this real-world WDN model, we only use the HV as the performance metric where $\mathbf{z}^r = (1.1, \cdots, 1.1)^T$. In particular, we normalize the objective functions before calculating the HV metric. From the box plots (with respect to 51 independent runs) shown in Figure 4.19, we can clearly see that our proposed C-TAEA shows better performance than the other five peer algorithms.

4.5 Further Study: Combination of A Changing Number of Objectives and Dynamic Constraints

In Chapter 3, DMOPs with a changing number of objectives are defined and discussed. DTAEA is proposed to solve this problem. In this chapter, C-TAEA is proposed for solving the MOPs with dynamic constraints. In this section, we try to discuss the combination of this two types of dynamics. We mainly focus on the ability of dynamic handling for two-archive evolutionary algorithms, in a general way. Based on the experimental result from the problems with a changing number of objectives and dynamic constraints, we try to analyse why two-archive algorithms can successfully handle different dynamics. It should be pointed out that although we have case studies for both DMOPs with a changing number of objectives and MOPs with dynamic constraints, we do not find any real-life cases which have included both dynamics in the optimization process.

4.5.1 Motivation

From the previous chapters, we find out that two archive algorithms have great ability to overcome dynamic situations. With the help of CA and DA, DTAEA can handle the expansion and contraction of the dimension of the PS/PF manifold dealing when dealing with problems with a changing number of objectives; C-TAEA can track the moving feasible regions by maintaining infeasible solutions in DA when dealing with problems with dynamic constraints. Here we are going to discuss the situations for DMOPs with a changing number of objectives and dynamic constraints.

Although the features for both DMOPs with a changing number of objectives and MOPs with dynamic constraints have been discussed in Chapter 3 and Chapter 4, however the features for DMOPs with a changing number of objectives together with dynamic constraints are different. Possible types of changes are listed in Table 4.9.

Type-C and Type-D are problems with a changing number of objectives, which have been discussed in Chapter 3. Type-E are MOPs with dynamic constraints; Type-F are

Table 4.9: Problems with six types of changes

Num of Obj.	Constraints	
	change	not change
increases	Type-A	Type-C
decreases	Type-B	Type-D
not change	Type-E	Type-F

MOPs with static constraints, which are mentioned in Chapter 4. In this section, we only focus on Type-A and Type-B.

In this section, we denoted the PF of the original problem without constraints as PF_o, and the PF for the problem with constraint as PF_c. Assumed that the change happens at time t, the fronts before the change are denoted as PF_o^{t-1} and PF_c^{t-1}, the fronts after the change are denoted as PF_o^t and PF_c^t.

Features for Type-A problems

For problems in this type, the constraints are changing when the number of objectives increases. When the number of objective increases and the constraints changes, a solution $\mathbf{x} \in PF_o^{t-1}$ which will still be on the PF_o^t (the conclusion from Chapter 3).

$$\mathbf{x} \in PF_o^{t-1} \Rightarrow \mathbf{x} \in PF_o^t \tag{4.36}$$

If $PF_c^{t-1} \subseteq PF_o^{t-1}$, we can get:

$$\mathbf{x} \in PF_c^{t-1} \Rightarrow \mathbf{x} \in PF_o^t \tag{4.37}$$

This means a well approximated population will be still on the PF without constraints (PF_o^t). Some solutions may become infeasible after the changes.

Features for Type-B problems

For problems in this type, the constraints are changing when the number of objectives decreases. When the number of objective decreases and the constraints change in the same time, there exist solutions $\mathbf{x}^1 \in PF_o^{t-1}$ will still be on the PF_o^t, while other solutions $\mathbf{x}^2 \in PF_o^{t-1}$ are dragged into the search space (the conclusion from Chapter 3). If $PF_c^{t-1} \subseteq PF_o^{t-1}$, we can get:

$$\exists \mathbf{x}, \mathbf{x} \in PF_c^{t-1} \wedge \mathbf{x} \in PF_o^t \tag{4.38}$$

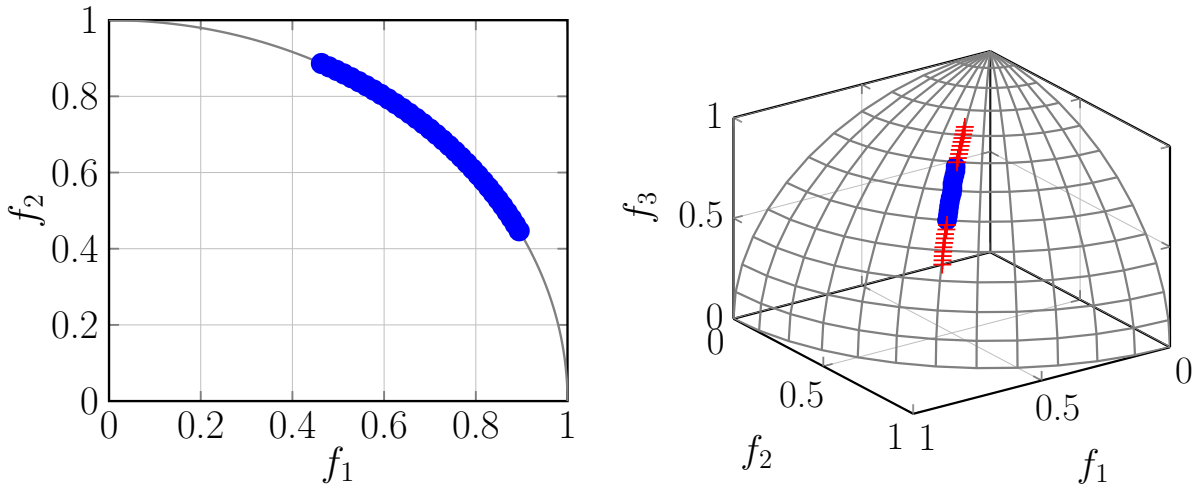
Simple Examples

In order to understand the challenges in Type-A and Type-B, we demonstrate a simple experiment in this part. In this experiment the C2DTLZ2 [64] is the benchmark problem and constrained multi-objective EA based on decomposition (CMOEA/D) [11] is the baseline algorithm.

Situation for Type-A problems: Here we demonstrate the C2DTLZ2 with following situation: at generation 200, the number of objectives for C2DTLZ2 is changing from 2 to 3, and the r is changing from 0.3 to 0.2 in the same time.

First, a well approximation to the PF of two-objective CDTLZ2 is obtained from a 200 generations' run with the CMOEA/D. The result is shown in Figure 4.20(a). Solutions are well-distributed on the whole PF_c^{t-1}. Then, the underlying problem changes to a three-objective instance and the r changes to 0.2. Figure 4.20(b) shows the distribution of the population gained after re-evaluating the objective values. In this figure, the blue $\cdot$ represents the feasible solutions and the red + represents the infeasible ones. We can find out that all of the population still on the PF_o^t, some solutions become infeasible. Also the diversity is not satisfactory any longer after the change as the population locates on a curve.

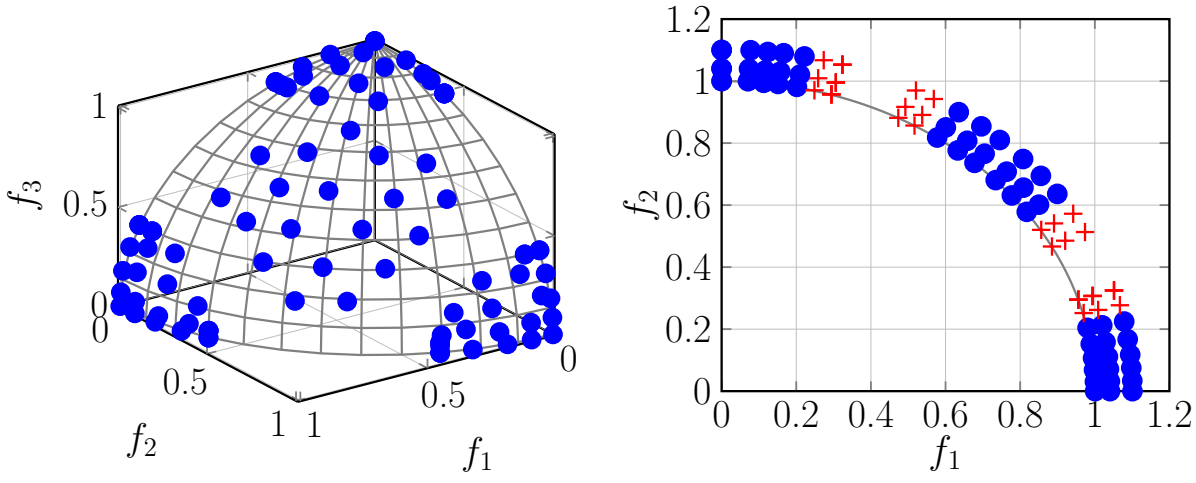
Situation for Type-B problems: Here we demonstrate the C2DTLZ2 with following situation: at generation 200, the number of objectives for C2DTLZ2 is changing from 3 to 2, and the r is changing from 0.2 to 0.3 in the same time.



(a) Approximation in the 2-objective case (b) Scatter plot after increasing an objective

Figure 4.20: Comparison of population distributions for Type-A

CMOEA/D is run on a three-objective C2DTLZ2 with $r = 0.3$ instance for 200 generations. A reasonably good approximation to the PF is obtained which is shown in Figure 4.21(a). Then, the underlying problem changes to a two-objective with $r = 0.2$ in the same time. Figure 4.21(b) shows the corresponding population distribution after re-evaluating the objective values. Only a few of solutions are on the PF_c^t, others are either drifted from the PF_o^t or infeasible.



(a) Approximation in the 3-objective case. (b) Scatter plot after descreasing an objective.

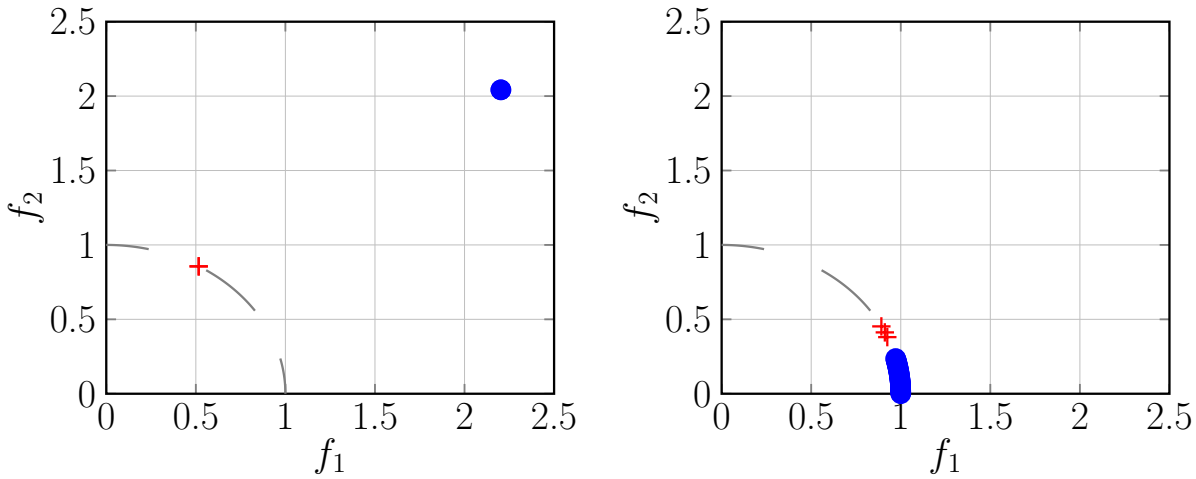
Figure 4.21: Comparison of population distributions for Type-B.

How to deal with the infeasible solutions are one of the major challenges for prob-

lems with a changing number of objectives and dynamic constraints. As most of the algorithms prefer feasible solutions than infeasible one, algorithms (e.g. CMOEA/D) will discard all the infeasible solutions and replace with some new generated feasible solutions after the changes, which obviously is not a good decision. The following Figure 4.22(a) shows a simple example. Most of the existing algorithms, such as CNSGA-II/CNSGA-III, MOEA/D, MOEA/DD will discard the infeasible solutions (red + in Figure 4.21), but choose the feasible solutions (blue • in Figure 4.21). This is obviously a wrong decision, as the infeasible solutions are closer to the PF than the new generated feasible solutions.

Another example can be found in Figure 4.22(b). Although all the feasible solutions are well-converged, they are crowding in one feasible region and no single solution is in the other two feasible regions. However, if the algorithms do not maintain infeasible solutions shown in the figure, it is hard to explore the whole search space.

Generally speaking, the idea of "feasible solutions are better than an infeasible one" is wrong in this case. A smarter algorithm is needed to better utilise the infeasible solutions.



(a) Replacing the more valuable solution with the less valuable one

(b) Preventing solutions to cross the gap

Figure 4.22: Examples for wrong decisions of infeasible solutions handling

Table 4.10: Mathematical Definitions of Dynamic Multi-Objective Benchmark Problems

Problem Instance	Definition	Domain
F1	$f_1 = (1+g)0.5\prod_{i=1}^{m(t)-1} x_i$ $f_{j=2:m(t)-1} = (1+g)0.5(\prod_{i=1}^{m(t)-j} x_i)(1-x_{m(t)-j+1})$ $f_{m(t)} = (1+g)0.5(1-x_1)$ $g = 100[n-m(t)+1+\sum_{i=m(t)}^{n}((x_i-0.5)^2-\cos(20\pi(x_i-0.5)))]$	$[0,1]$
F2	$f_1 = (1+g)0.5\prod_{i=1}^{m(t)-1}\cos(x_i\pi/2)$ $f_{j=2:m(t)-1} = (1+g)0.5(\prod_{i=1}^{m(t)-j}\cos(x_i\pi/2))(\sin(x_{m(t)-j+1}\pi/2))$ $f_{m(t)} = (1+g)\sin(x_1\pi/2)$ $g = \sum_{i=m(t)}^{n}(x_i-0.5)^2$	$[0,1]$
F3	as F2, except g is replaced by the one from F1	$[0,1]$
F4	as F2, except x_i is replaced by x_i^α, where $i \in \{1,\cdots,m(t)-1\}, \alpha > 0$	$[0,1]$

4.5.2 Benchmark Problems

The problems selected as test problems are based on the problems proposed in Chapter 3, but with dynamic constraints.

The constraints for F1 are:

$$c_j(\mathbf{x}) = cos(a\pi x_j) > b, \forall_j = 1, ..., m \tag{4.39}$$

$$c_{m+1}(\mathbf{x}) = cos(a\pi g(\mathbf{x}_m)) > b. \tag{4.40}$$

The constraint for F2 is:

$$c(\mathbf{x}) = \max\{\max_{i=1}^{m}[(f_i(\mathbf{x})-1)^2 + \sum_{j=1,j\neq i}^{m}(f_j^2 - r^2)], [\sum_{i=1}^{m}(f_i(\mathbf{x})-r_i)^2 - r^2]\} \tag{4.41}$$

in which $r_i = \frac{\cos(0.25\pi(t+i)}{\sqrt{m}}$.

The constraint for F3 is:

$$c(\mathbf{x}) = (\sum_{i=1}^{m} f_i(\mathbf{x})^2 - 16)(\sum_{i=1}^{m} f_i(\mathbf{x})^2 - r^2) \geq 0. \tag{4.42}$$

in which $r = 8 + 2\cos(0.25\pi t)$.

The constraint for F4 is:

$$c_j(\mathbf{x}) = \frac{f_j^2}{4} + \sum_{i=1, i\neq j}^{m} f_i(\mathbf{x})^2 - 1 \geq 0, \forall_j = 1, ..., m. \tag{4.43}$$

in which $r = 4(1 + 0.5\cos(0.25\pi t))$

4.5.3 Implementation

The DTAEA is adapted for the constrained problems, named as C-DTAEA. The C-DTAEA is a combination of C-TAEA and DTAEA. C-DTAEA has a different reconstruction mechanism compared to C-TAEA and DTAEA.

Reconstruction Mechanism

The basic idea of reconstruction is relocating or replacing solutions in CA and DA. Since the increasing and decreasing number of objectives have similar challenges, the reconstruction mechanisms are the same.

Some solutions from the convergence of the population is not affected when increasing or decrease the number of objectives, while some others will become infeasible, but still close to the PF. Also the population diversity is unsatisfying. As a result, all feasible solutions will be constructed as the new CA. As the new CA is not full, the CA will be filled up with polynomial mutated [115] solutions which generated from some selected competitive non-dominated solutions in the new CA. In particular, the solutions used for mutation are chosen by a binary tournament selection, where the non-dominated solution located in a less density area is preferred. The other solutions will be kept in the DA. As the new DA is not full, the randomly generated solutions will be filled up with. In particular, the canonical Latin hypercube sampling (LHS) method [114] are employed to uniformly sample N random solutions from the decision space. The pseudo-code of this reconstruction mechanism is given in Algorithm 14.

The *FeasibleSolutionSelection* picks all the feasible solutions from the set, and *Infea-*

Algorithm 14: Reconstruction Mechanism

Input: The CA,DA before changes, population size N
Output: CA, DA

```
P ← CA ⋃ DA;
CA ← FeasibleSolutionSelection(P);
CA ← FeasibleSolutionSelection(P);
if |CA| > N then
    updateCA(CA);                                   // update CA as DTAEA
if |DA| > N then
    CVSection(DA);                  // choose solutions with smaller CV
while |DA| < N do
    DA ← RandomSolution();
while |CA| < N do
    x^c ← BinaryTournamentSelection(CA);
    x^m ← PolynomialMutation(x^c);
    CA ← CA ⋃ {x^m};
return CA and DA
```

sibleSolutionSelection picks all the infeasible ones. The *BinaryTournamentSelection* is described in Section 4.2.3.

Algorithm 15: Reconstruction Mechanism

Input: The last CA before changes
Output: CA, DA

```
CA ← FeasibleSolutionSelection(P);    // choose all feasible solutions
DA ← P \ CA;
while |CA| < N do
    x^c ← BirnaryTournamentSelection(CA);
    x^m ← PolynomialMutation(x^c);
    CA ← CA ⋃ {x^m};
Use the LHS to sample N − |DA| random solutions to fill the DA;
return CA and DA
```

4.5.4 Results

This section contains the experiment. The test problems are F1 to F4 with dynamic constraints The comparing algorithms are selected as followed:

- *C-DNSGA-II*: Currently there are no algorithms are proposed for constrained and

dynamic problems. However, the technique of constraints handle mentioned by Jain et al. [11] is compatible with the dynamic handling technique proposed by Deb et al. [9]. In particular, when there is no changes, the algorithm select solutions based on the constraint-domination principle (CDP). When change happens, some population members are replaced with randomly generated solutions.

- *C-DNSGA-III*: NSGA-III is a recently developed NSGA-II variant which can handle problems with more than three objectives. The only difference lies in the density estimation method which uses several pre-defined reference points to specify niches in the objective space. Solutions in the less crowded niche has a higher priority to survive to the next generation. The same as C-DNSGA-II, both technique for dynamic handling and constraint handling can be added to NSGA-III.

- *C-MOEA/D-KF*: This is a recently proposed prediction-based strategy that employs a linear discrete time Kalman Filter to model the movements of the PS in a dynamic environment. Thereafter, this model is used to predict the new location of the PS when a change occurs. Empirical results in [96] has shown that MOEA/D-KF is very competitive for the dynamic optimization and it outperforms the other state-of-the-art predictive strategies, e.g., Zhou et al. [91] and Wu et al. [94]. By applying the constraint-domination principle, the MOEA/D-KF is adapted to a constrained version.

- *C-MOEA/DD*: This algorithm is well known for constraints handling. MOEA/DD [75] update solutions in a special way. The algorithm unified paradigm, which combines dominance- and decomposition-based approaches, which leads a better result in both unconstrained and constrained problems. Noted that C-MOEA/DD have no dynamic handling technique.

- *C-MOEA/D*: This is a representative of the decomposition-based EMO methods. By adopting the constraint-dominated principle, MOEA/D is adjusted for the constraint handling. Noted that C-MOEA/DD have no dynamic handling technique.

Table 4.11: Performance Comparisons of C-DTAEA and the Other Algorithms on MIGD Metric

problem	τ	C-NSGA-II	C-NSGA-III	C-MOEA/D-KF	C-MOEA/DD	C-MOEA/D	C-DTAEA
F1	25	1.0071E+1(1.46E-3)†	8.9533E+0(5.62E-2)†	8.1428E+0(6.55E-2)†	8.2773E+0(5.92E-3)†	8.5256E+0(7.35E-2)†	**3.6126E-1(9.58E-2)**
	50	8.6582E+0(6.25E-2)†	8.1343E+0(5.21E-2)†	8.0828E+0(5.56E-2)†	8.1123E+0(2.49E-2)†	8.1885E+0(4.89E-3)†	**3.2184E-2(1.30E-2)**
	100	9.5694E+0(4.62E-2)†	8.0556E+0(2.44E-2)†	8.0527E+0(9.99E-2)†	8.1016E+0(1.25E-3)†	8.2846E+0(9.12E-2)†	**3.4186E-2(2.30E-2)**
	200	9.4967E+0(7.34E-2)†	8.0508E+0(4.06E-2)†	8.0520E+0(3.30E-2)†	8.1015E+0(5.43E-2)†	8.1057E+0(8.90E-2)†	**3.5583E-2(1.99E-2)**
F2	25	6.6014E-1(5.79E-2)†	1.0045E-2(7.59E-2)†	6.7846E-2(6.88E-2)†	1.3429E-1(6.59E-3)†	2.4031E-1(1.07E-2)†	**3.2983E-3(8.62E-2)**
	50	2.6556E-1(4.50E-2)†	1.8870E-1(6.95E-2)†	7.8131E-2(3.21E-2)†	1.9003E-1(8.11E-2)†	8.8686E-2(2.78E-4)†	**2.2480E-2(3.45E-2)**
	100	1.3802E-2(1.24E-2)†	5.3261E-2(4.92E-2)†	7.9250E-2(9.97E-3)†	1.2617E-1(9.06E-2)†	1.2278E-2(3.34E-2)†	**8.6564E-3(3.00E-2)**
	200	6.4051E-1(3.83E-2)†	1.2598E-1(4.64E-2)†	1.2100E-1(4.34E-2)†	2.5246E-2(3.82E-2)†	9.4107E-2(2.42E-2)†	**7.5586E-3(3.39E-2)**
F3	25	1.4446E+0(1.43E-2)†	1.3192E-1(4.71E-2)†	2.4041E-1(3.40E-2)†	2.4538E-1(4.77E-2)†	1.3510E-1(7.81E-2)	**1.3507E-3(2.88E-2)**
	50	2.7305E-1(6.47E-2)†	2.1286E-1(3.09E-2)†	7.4913E-2(9.45E-2)†	1.9044E-1(8.87E-2)†	2.1093E-2(9.20E-2)†	**2.5082E-3(7.21E-2)**
	100	1.0523E+0(8.30E-2)†	4.7857E-2(1.20E-2)†	2.8097E-2(9.94E-2)†	6.3608E-2(9.41E-3)†	2.2682E-2(4.26E-2)†	**1.7271E-2(8.48E-2)**
	200	5.6142E-1(1.76E-3)†	1.4366E-1(5.35E-2)†	1.8386E-1(4.81E-2)†	4.9403E-2(4.72E-2)†	1.7194E-2(9.11E-2)†	**8.5840E-3(4.95E-2)**
F4	25	1.4446E+0(1.43E-2)†	1.3512E-2(4.71E-2)	2.4041E-2(3.40E-2)†	2.4538E-2(4.77E-2)†	1.5510E-2(7.81E-3)†	**1.3507E-2(2.88E-2)**
	50	2.7305E-1(6.47E-2)†	2.1286E-2(3.09E-2)	7.4913E-2(9.45E-2)†	1.9044E-1(8.87E-2)†	2.7093E-3(9.20E-2)†	**2.1282E-2(7.21E-2)**
	100	1.0523E+0(8.30E-2)†	4.7857E-2(1.20E-2)†	2.8097E-2(9.94E-2)†	6.3608E-2(9.41E-3)†	2.2682E-1(4.26E-2)†	**1.7271E-2(8.48E-2)**
	200	5.6142E-1(1.76E-3)†	1.4366E-1(5.35E-2)†	**8.5786E-3(4.81E-2)**	8.1786E-3(4.72E-2)	8.5586E-3(9.11E-2)	8.5840E-3(4.95E-2)

Each algorithm is independently run 51 times on each problem instance. 300 generations are given to each algorithm before the first change. In other words, the first change occurs after the first 300 generations. To have a statistically sound conclusion, we use the Wilcoxon rank sum test at the 5% significance level in the comparisons.

The time varying number of objectives $m(t)$ as follows:

$$m(t) = \begin{cases} 3, & t = 1 \\ m(t-1) + 1, & t \in [2, 5] \\ m(t-1) - 1, & t \in [6, 10] \end{cases} \tag{4.44}$$

where $t \in \{1, \cdots, 10\}$ is a discrete time. To investigate the performance of different algorithms under various frequencies of change, τ_t is set as 25, 50, 100, 200, respectively. Table 4.11 give the median and the interquartile range (IQR) of the corresponding metric values obtained by different algorithms under various circumstances. In particular, the best metric values are highlighted in the bold face with a gray background. We set its change frequency as $\tau_{\bar{t}} = 25$ and change severity as $n_{\bar{t}} = 25$. The following Table 4.11 gives all the results.

The experiment results for C-DTAEA is overwhelmingly satisfying. On 16 cases, there is only one single case that the performance of C-DTAEA is poorer than other algorithms. The general reasons for that can be listed as followed:

- The two-archive EA is suitable for constrained problems. As the DA ignores the constraints, the solution of the DA can explore the whole searching and find out any potential feasible regions. The DA can go through any CV local optimum as the DA do not consider the CV.

- The two-archive EA is especially suitable for dynamic constrained problems. When handling dynamic constrained problems, such as F1 and F2, the changing constraints actually makes part of infeasible region becomes feasible, while some feasible region becomes infeasible. For algorithms based on two archives, the DA is the complement of the CA when both archives are converged to the PF. In this case, the two-archive algorithm can find out the new feasible solutions simply moving solutions from DA to CA.

- C-DTAEA is the only algorithm that can handle DMOPs with a changing number of objectives in the comparison. C-DTAEA is the only algorithm can achieve a relatively converged population.

In the cases of F1 and F2, as mention above, when the constraints change, it will bring troubles for other algorithms as those algorithms will discard solutions which turn from feasible to infeasible, then fill the population with randomly generated feasible solutions. However, C-DTAEA simple put infeasible solutions into DA. When these solutions become feasible again, C-DTAEA does not need to process the search.

In the case of F4, C-DTAEA has to face some failures. As the infeasible region covers the whole PF, the DA is going into the infeasible region, which will not help. Even worse, as the solutions in DA step into the infeasible region, the mating mechanism will always choose solutions from DA. When the edge of feasible regions moving towards the opposite direction of the PF, the force pushing solutions out from the infeasible region is limited.

However, C-DTAEA gains advantages from handling situations with a changing number of objectives, which C-DTAEA still is the best in most of the case.

All in all, the two-archive EA is good at solving constrained problems, experientially those with dynamic constraints. Combining the advantages from the changing number of objectives and the dynamic constraints, the C-DTAEA wins 15 out 16 cases, which is even better than the DTAEA can achieve when dealing with the DMOPs with a changing number of objectives.

4.5.5 Conclusion

In this section, we extend the DMOPs with a changing number of objectives to a more complex case: DMOPs with a changing number of objectives and dynamic constraints. The changes of constraints mix up with the changes of changing number of objectives bring new features. According to the features, the C-DTAEA is designed. The C-DTAEA is adapted from the DTAEA, with a different reconstructing mechanism. The C-DTAEA shows excellent performance when dealing with DMOPs with a changing number of objectives an dynamic constraints. That is because the algorithms based on the two-archive EA framework are good at solving constrained problems, especially those with dynamic constraints.

CHAPTER 5

CONCLUSION

In this chapter, a global view on the contributions of this thesis is presented. A list of all the research findings and conclusions are presented in Section 5.1. Meanwhile, we will show how we handle the challenges mentioned at the beginning of the thesis. Section 5.2 discusses the future works.

5.1 Summary of Results

Dynamic multi-objective optimization problem with a changing number of objectives poses great challenges to evolutionary algorithms. Instead of changing the shape or position, the dimension of the Pareto-optimal front/set manifold changes. The dynamic handling techniques can hardly handle this challenge as these techniques are just designed for tracking the moving front, rather than exploring an expanding or contracting one.

In Chapter 3, we propose our algorithms DTAEA for the DMOPs with a changing number of objectives. In Chapter 4 we propose our algorithms C-TAEA for MOPs with time-dependent constraints. In the following paragraphs, we will provide a summary of the research results for these chapters.

In Chapter 3, two main characteristics for the DMOPs with a changing number of objectives are found after the analysis: 1) the Pareto-optimal front/set before the change is

a subset of the one after the number of objectives increases, and vice versa; and 2) instead of changing the position or the shape of the original Pareto-optimal front/set, increasing or decreasing the number of objectives usually results in the expansion and contraction of the dimension of the Pareto-optimal front/set manifold. Based on these characteristics, we adopt a two-archive evolutionary algorithm for this problem, named as DTAEA. Also we adapt the DTLZ series for the DMOPs with a changing number of objectives. The DTAEA have performed very well in problem handling. In the further studies, we have discussed the contributions of each component in DTAEA. Then we find out something more about the DTAEA. First, DTAEA can work under high-frequency changes, but changing too frequently (5 generations between two changes) will cause a failure. Second, problems with partitioned decision variables have no impact on DTAEA. Finally, we propose a new idea to solve the many-objective problem. The optimization process starts with two-objectives, and once the good approximate Pareto-optimal population is achieved, one objective will be added. This method may work in some cases, but further development is still needed.

In Chapter 4, we adapt DTAEA for MOPs with dynamic constraints (DCMOPs), named C-TAEA. The situation of dynamic constraints considered in this thesis only includes time-dependent constraints. We first analyse the challenges for the DCMOPs. Then we propose self-designed test problem suites, both for static CMOPs and dynamic CMOPs. And from the experiment, we find out an important conclusion that infeasible solutions sometimes are more valuable than those feasible ones, in terms of speeding up the convergent speed and enhancing the diversity. We adapt C-TAEA to real-life cases: water distribution network problems (WDN). C-TAEA performs well on WDN. Finally, we combine the two types of dynamics mentioned in this thesis. We try to solve a DMOP with a changing number of objectives and dynamic constraints.

5.2 Future Work

There are many directions for future research:

- There are many different ways that the number of objectives may change in practice. This thesis considered only some simple cases. In the future, we would like to study the cases where the number of objectives changed may be greater than one and the cases where changes are a mixture of increases and decreases.

- The changes in both the number of objectives and objective functions themselves are an interesting topic for future research.

- It is interesting to investigate the relationships between a changing number of objectives and changing preferences with a fixed number of objectives.

Here we propose two directions from our current research interest.

- **Solving many-objectives optimization problem as DMOPs with a changing number of objectives.** It is well known that exploring the high dimension searching space poses great challenges for EA. It is possible to solve this problem as a DMOP with a changing number of objectives. Although the challenges for this method are huge, from our simple experiments the DTAEA shows promises in solving the many-objectives problem.

- **Modelling real-life problems as a DMOPs with a changing number of objectives.** Many real-life applications can be considered as a DMOP with a changing number of objectives. The scheduling problem on RUP mentioned in this thesis is a good example. Another example is the water distribution network problems mentioned in Chapter 4. From our experiments, we have already found that considering problems as DMOPs with a changing number of objectives have many advantages.

LIST OF REFERENCES

[1] G. Pellegrino and F. Cupertino, "Fea-based multi-objective optimization of ipm motor design including rotor losses," in *Energy Conversion Congress and Exposition (ECCE), 2010 IEEE*. IEEE, 2010, pp. 3659–3666.

[2] M. Yu, J. Zhang, and W. Zhang, "Multi-objective optimization design method of the high-speed train head," *Journal of Zhejiang University SCIENCE A*, vol. 14, no. 9, pp. 631–641, 2013.

[3] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[4] E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," in *Parallel Problem Solving from Nature - PPSN VIII, 8th International Conference, Birmingham, UK, September 18-22, 2004, Proceedings*, 2004, pp. 832–842.

[5] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[6] K. Praditwong and X. Yao, "A new multi-objective evolutionary optimisation algorithm: The two-archive algorithm," in *CIS'06: Proc. of the 2006 Computational Intelligence and Security*, 2006, pp. 95–104.

[7] H. Wang, L. Jiao, and X. Yao, "Two_arch2: An improved two-archive algorithm for many-objective optimization," *IEEE Trans. Evolutionary Computation*, vol. 19, no. 4, pp. 524–541, 2015.

[8] A. Zhou, Y. Jin, and Q. Zhang, "A population prediction strategy for evolutionary dynamic multiobjective optimization," *IEEE Trans. Cybernetics*, vol. 44, no. 1, pp. 40–53, 2014. [Online]. Available: https://doi.org/10.1109/TCYB.2013.2245892

[9] K. Deb, U. B. Rao, and S. Karthik, "Dynamic multi-objective optimization and decision-making using modified NSGA-II: A case study on hydro-thermal power scheduling," in *EMO'07: Proc. of the 4th International Conference on Evolutionary Multi-Criterion Optimization*, 2007, pp. 803–817.

[10] A. Gunasekaran, C. Patel, and R. E. McGaughey, "A framework for supply chain performance measurement," *International journal of production economics*, vol. 87, no. 3, pp. 333–347, 2004.

[11] H. Jain and K. Deb, "An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part II: handling constraints and extending to an adaptive approach," *IEEE Trans. Evolutionary Computation*, vol. 18, no. 4, pp. 602–622, 2014.

[12] M. Farina, K. Deb, and P. Amato, "Dynamic multiobjective optimization problems: test cases, approximations, and applications," *IEEE Transactions on evolutionary computation*, vol. 8, no. 5, pp. 425–442, 2004.

[13] C. Raquel and X. Yao, "Dynamic multi-objective optimization: A survey of the state-of-the-art," vol. 490, 01 2013, pp. 85–106.

[14] T. T. Nguyen, "Continuous dynamic optimisation using evolutionary algorithms," Ph.D. dissertation, University of Birmingham, UK, 2011. [Online]. Available: http://etheses.bham.ac.uk/1296/

[15] L. Huang, I. H. Suh, and A. Abraham, "Dynamic multi-objective optimization based on membrane computing for control of time-varying unstable plants," *Inf. Sci.*, vol. 181, no. 11, pp. 2370–2391, 2011.

[16] S. Majumdar, K. Mitra, and G. Sardar, "Kinetic analysis and optimization for the catalytic esterification step of ppt polymerization," *Macromolecular Theory and Simulations*, vol. 14, no. 1, pp. 49–59, 2005.

[17] S. Manos, L. Poladian, P. J. Bentley, and M. Large, "Photonic device design using multiobjective evolutionary algorithms," in *Evolutionary Multi-Criterion Optimization, Third International Conference, EMO 2005, Guanajuato, Mexico, March 9-11, 2005, Proceedings*, 2005, pp. 636–650. [Online]. Available: https://doi.org/10.1007/978-3-540-31880-4_44

[18] R. Farmani, D. Savic, and G. Walters, "Evolutionary multi-objective optimization in water distribution network design," *Engineering Optimization*, vol. 37, no. 2, pp. 167–183, 2005.

[19] S. Poles, Y. Fu, and E. Rigoni, "The effect of initial population sampling on the convergence of multi-objective genetic algorithms," in *Multiobjective programming and goal programming.* Springer, 2009, pp. 123–133.

[20] M. D. McKay, R. J. Beckman, and W. J. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 42, no. 1, pp. 55–61, 2000. [Online]. Available: https://doi.org/10.1080/00401706.2000.10485979

[21] A. Szöllös, M. Smíd, and J. Hájek, "Aerodynamic optimization via multi-objective micro-genetic algorithm with range adaptation, knowledge-based reinitialization, crowding and epsilon-dominance," *Advances in Engineering Software*, vol. 40, no. 6, pp. 419–430, 2009. [Online]. Available: https://doi.org/10.1016/j.advengsoft.2008.07.002

[22] R. Chen, K. Li, and X. Yao, "Dynamic multi-objectives optimization with a changing number of objectives," *CoRR*, vol. abs/1608.06514, 2016. [Online]. Available: http://arxiv.org/abs/1608.06514

[23] M. A. Oliver and R. Webster, "Kriging: a method of interpolation for geographical information systems," *International Journal of Geographical Information Science*, vol. 4, no. 3, pp. 313–332, 1990. [Online]. Available: https://doi.org/10.1080/02693799008941549

[24] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, 2000. [Online]. Available: https://doi.org/10.1162/106365600568202

[25] J. R. Kalagnanam and U. M. Diwekar, "An efficient sampling technique for off-line quality control," *Technometrics*, vol. 39, no. 3, pp. 308–319, 1997.

[26] A. Jamali, A. Hajiloo, and N. Nariman-Zadeh, "Reliability-based robust pareto design of linear state feedback controllers using a multi-objective uniform-diversity genetic algorithm (MUGA)," *Expert Syst. Appl.*, vol. 37, no. 1, pp. 401–413, 2010. [Online]. Available: https://doi.org/10.1016/j.eswa.2009.05.048

[27] S. Z. Martínez and C. A. C. Coello, "A proposal to hybridize multi-objective evolutionary algorithms with non-gradient mathematical programming techniques," in *Parallel Problem Solving from Nature - PPSN X, 10th International Conference Dortmund, Germany, September 13-17, 2008, Proceedings*, 2008, pp. 837–846. [Online]. Available: https://doi.org/10.1007/978-3-540-87700-4_83

[28] B. L. Miller and D. E. Goldberg, "Genetic algorithms, tournament selection, and the effects of noise," *Complex Systems*, vol. 9, no. 3, 1995. [Online]. Available: http://www.complex-systems.com/abstracts/v09_i03_a02.html

[29] E. Zitzler and L. Thiele, "An evolutionary algorithm for multiobjective optimization: The strength pareto approach," *TIK-report*, vol. 43, 1998.

[30] M. Kim, T. Hiroyasu, M. Miki, and S. Watanabe, "SPEA2+: improving the performance of the strength pareto evolutionary algorithm 2," in *Parallel Problem Solving from Nature - PPSN VIII, 8th International Conference, Birmingham, UK, September 18-22, 2004, Proceedings*, 2004, pp. 742–751. [Online]. Available: https://doi.org/10.1007/978-3-540-30217-9_75

[31] J. Holland and D. Goldberg, "Genetic algorithms in search, optimization and machine learning," *Massachusetts: Addison-Wesley*, 1989.

[32] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[33] A. J. Nebro, F. Luna, E. Alba, B. Dorronsoro, J. J. Durillo, and A. Beham, "Abyss: Adapting scatter search to multiobjective optimization," *IEEE Trans. Evolutionary Computation*, vol. 12, no. 4, pp. 439–457, 2008. [Online]. Available: https://doi.org/10.1109/TEVC.2007.913109

[34] A. Elhossini, S. Areibi, and R. D. Dony, "Strength pareto particle swarm optimization and hybrid EA-PSO for multi-objective optimization," *Evolutionary Computation*, vol. 18, no. 1, pp. 127–156, 2010. [Online]. Available: https://doi.org/10.1162/evco.2010.18.1.18105

[35] D. W. Corne, N. R. Jerram, J. D. Knowles, and M. J. Oates, "Pesa-ii: Region-based selection in evolutionary multiobjective optimization," in *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*. Morgan Kaufmann Publishers Inc., 2001, pp. 283–290.

[36] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[37] Y. Qi, X. Ma, F. Liu, L. Jiao, J. Sun, and J. Wu, "MOEA/D with adaptive weight adjustment," *Evolutionary Computation*, vol. 22, no. 2, pp. 231–264, 2014. [Online]. Available: https://doi.org/10.1162/EVCO_a_00109

[38] Q. Zhang, W. Liu, and H. Li, "The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances," in *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2009, Trondheim, Norway, 18-21 May, 2009*, 2009, pp. 203–208. [Online]. Available: https://doi.org/10.1109/CEC.2009.4982949

[39] A. Zhou and Q. Zhang, "Are all the subproblems equally important? resource allocation in decomposition-based multiobjective evolutionary algorithms," *IEEE Trans. Evolutionary Computation*, vol. 20, no. 1, pp. 52–64, 2016. [Online]. Available: https://doi.org/10.1109/TEVC.2015.2424251

[40] T. Chiang and Y. Lai, "MOEA/D-AMS: improving MOEA/D by an adaptive mating selection mechanism," in *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2011, New Orleans, LA, USA, 5-8 June, 2011*, 2011, pp. 1473–1480. [Online]. Available: https://doi.org/10.1109/CEC.2011.5949789

[41] I. Giagkiozis, R. C. Purshouse, and P. J. Fleming, "Generalized decomposition," in *Evolutionary Multi-Criterion Optimization - 7th International Conference, EMO 2013, Sheffield, UK, March 19-22, 2013. Proceedings*, 2013, pp. 428–442. [Online]. Available: https://doi.org/10.1007/978-3-642-37140-0_33

[42] ——, "Generalized decomposition and cross entropy methods for many-objective optimization," *Inf. Sci.*, vol. 282, pp. 363–387, 2014. [Online]. Available: https://doi.org/10.1016/j.ins.2014.05.045

[43] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated pareto sets, MOEA/D and NSGA-II," *IEEE Trans. Evolutionary Computation*, vol. 13, no. 2, pp. 284–302, 2009.

[44] K. Li, Á. Fialho, S. Kwong, and Q. Zhang, "Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evolutionary Computation*, vol. 18, no. 1, pp. 114–130, 2014. [Online]. Available: https://doi.org/10.1109/TEVC.2013.2239648

[45] L. Ke, Q. Zhang, and R. Battiti, "MOEA/D-ACO: A multiobjective evolutionary algorithm using decomposition and antcolony," *IEEE Trans. Cybernetics*, vol. 43, no. 6, pp. 1845–1859, 2013. [Online]. Available: https://doi.org/10.1109/TSMCB.2012.2231860

[46] K. Li, Q. Zhang, S. Kwong, M. Li, and R. Wang, "Stable matching-based selection in evolutionary multiobjective optimization," *IEEE Trans. Evolutionary Computation*, vol. 18, no. 6, pp. 909–923, 2014. [Online]. Available: https://doi.org/10.1109/TEVC.2013.2293776

[47] K. Li, S. Kwong, Q. Zhang, and K. Deb, "Interrelationship-based selection for decomposition multiobjective optimization," *IEEE Trans. Cybernetics*, vol. 45, no. 10, pp. 2076–2088, 2015. [Online]. Available: https://doi.org/10.1109/TCYB.2014.2365354

[48] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints," *IEEE Trans. Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, 2014.

[49] E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," in *PPSN'04: Proc. of 8th International Conference on Parallel Problem Solving from Nature - PPSN VIII*, 2004, pp. 832–842.

[50] N. Beume, B. Naujoks, and M. T. M. Emmerich, "SMS-EMOA: multiobjective selection based on dominated hypervolume," *European Journal of Operational Research*, vol. 181, no. 3, pp. 1653–1669, 2007. [Online]. Available: https://doi.org/10.1016/j.ejor.2006.08.008

[51] T. Wang, R. Liaw, and C. Ting, "MOEA/D using covariance matrix adaptation evolution strategy for complex multi-objective optimization problems," in *IEEE Congress on Evolutionary Computation, CEC 2016, Vancouver, BC, Canada, July 24-29, 2016*, 2016, pp. 983–990. [Online]. Available: https://doi.org/10.1109/CEC.2016.7743896

[52] D. A. Van Veldhuizen and G. B. Lamont, "Evolutionary computation and convergence to a pareto front," in *Late breaking papers at the genetic programming 1998 conference*, 1998, pp. 221–228.

[53] K. Deb and S. Jain, "Running performance metrics for evolutionary multi-objective optimization," 2002.

[54] D. A. V. Veldhuizen and D. A. V. Veldhuizen, "Multiobjective evolutionary algorithms: Classifications, analyses, and new innovations," Evolutionary Computation, Tech. Rep., 1999.

[55] J. D. Knowles, L. Thiele, and E. Zitzler, "A tutorial on the performance assessment of stochastic multiobjective optimizers," *TIK-Report*, vol. 214, 2006.

[56] L. S. Batista, F. Campelo, F. G. Guimarães, and J. A. Ramírez, "Pareto cone ϵ-dominance: Improving convergence and diversity in multiobjective evolutionary algorithms," in *Evolutionary Multi-Criterion Optimization - 6th International Conference, EMO 2011, Ouro Preto, Brazil, April 5-8, 2011. Proceedings*, 2011, pp. 76–90. [Online]. Available: https://doi.org/10.1007/978-3-642-19893-9_6

[57] J. R. Schott, "Fault tolerant design using single and multicriteria genetic algorithm optimization." AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH, Tech. Rep., 1995.

[58] K. C. Tan, T. H. Lee, and E. F. Khor, "Evolutionary algorithms for multi-objective optimization: Performance assessments and comparisons," *Artif. Intell. Rev.*, vol. 17, no. 4, pp. 251–290, 2002. [Online]. Available: https://doi.org/10.1023/A:1015516501242

[59] S. Mostaghim and J. Teich, "A new approach on many objective diversity measurement," in *Practical Approaches to Multi-Objective Optimization, 7.-12. November 2004*, 2005. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2005/254

[60] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *IEEE Trans. Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[61] P. A. N. Bosman and D. Thierens, "The balance between proximity and diversity in multiobjective evolutionary algorithms," *IEEE Trans. Evolutionary Computation*, vol. 7, no. 2, pp. 174–188, 2003.

[62] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca, "Performance assessment of multiobjective optimizers: an analysis and review," *IEEE Trans. Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, 2003. [Online]. Available: https://doi.org/10.1109/TEVC.2003.810758

[63] G. L. Lizárraga, A. H. Aguirre, and S. B. Rionda, "G-metric: an m-ary quality indicator for the evaluation of non-dominated sets," in *Genetic and Evolutionary Computation Conference, GECCO 2008, Proceedings, Atlanta, GA, USA, July 12-16, 2008*, 2008, pp. 665–672. [Online]. Available: http://doi.acm.org/10.1145/1389095.1389227

[64] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable test problems for evolutionary multiobjective optimization," in *Evolutionary Multiobjective Optimization*, ser. Advanced Information and Knowledge Processing, A. Abraham, L. Jain, and R. Goldberg, Eds. Springer London, 2005, pp. 105–145.

[65] S. Huband, P. Hingston, L. Barone, and R. L. While, "A review of multiobjective test problems and a scalable test problem toolkit," *IEEE Trans. Evolutionary Computation*, vol. 10, no. 5, pp. 477–506, 2006. [Online]. Available: https://doi.org/10.1109/TEVC.2005.861417

[66] T. P. Runarsson and X. Yao, "Search biases in constrained evolutionary optimization," *IEEE Trans. Systems, Man, and Cybernetics, Part C*, vol. 35, no. 2, pp. 233–243, 2005. [Online]. Available: https://doi.org/10.1109/TSMCC.2004.841906

[67] C. M. Fonseca and P. J. Fleming, "Multiobjective optimization and multiple constraint handling with evolutionary algorithms. i. A unified formulation," *IEEE Trans. Systems, Man, and Cybernetics, Part A*, vol. 28, no. 1, pp. 26–37, 1998.

[68] C. A. Coello Coello and A. D. Christiansen, "Moses: A multiobjective optimization tool for engineering design," *Engineering Optimization*, vol. 31, no. 3, pp. 337–368, 1999.

[69] M. A. Jan and Q. Zhang, "Moea/d for constrained multiobjective optimization: Some preliminary experimental results," in *Computational Intelligence (UKCI), 2010 UK Workshop on*. IEEE, 2010, pp. 1–6.

[70] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff, "A reference vector guided evolutionary algorithm for many-objective optimization," *IEEE Trans. Evolutionary Computation*, vol. 20, no. 5, pp. 773–791, 2016.

[71] A. Oyama, K. Shimoyama, and K. Fujii, "New constraint-handling method for multi-objective and multi-constraint evolutionary optimization," *Transactions of the Japan Society for Aeronautical and Space Sciences*, vol. 50, no. 167, pp. 56–62, 2007.

[72] T. Takahama and S. Sakai, "Efficient constrained optimization by the ϵ constrained rank-based differential evolution," in *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2012, Brisbane, Australia, June 10-15, 2012*, 2012, pp. 1–8. [Online]. Available: https://doi.org/10.1109/CEC.2012.6256111

[73] S. Z. Martínez and C. A. C. Coello, "A multi-objective evolutionary algorithm based on decomposition for constrained multi-objective optimization," in *CEC'14: Proc. of the 2014 IEEE Congress on Evolutionary Computation*, 2014, pp. 429–436.

[74] M. Asafuddoula, T. Ray, and R. A. Sarker, "A decomposition-based evolutionary algorithm for many objective optimization," *IEEE Trans. Evolutionary Computation*, vol. 19, no. 3, pp. 445–460, 2015.

[75] K. Li, K. Deb, Q. Zhang, and S. Kwong, "An evolutionary many-objective optimization algorithm based on dominance and decomposition," *IEEE Trans. Evolutionary Computation*, vol. 19, no. 5, pp. 694–716, 2015.

[76] W. Ning, B. Guo, Y. Yan, X. Wu, J. Wu, and D. Zhao, "Constrained multi-objective optimization using constrained non-dominated sorting combined with an improved hybrid multi-objective evolutionary algorithm," *Engineering Optimization*, vol. 49, no. 10, pp. 1645–1664, 2017.

[77] Y. G. Woldesenbet, G. G. Yen, and B. G. Tessema, "Constraint handling in multiobjective evolutionary optimization," *IEEE Trans. Evolutionary Computation*, vol. 13, no. 3, pp. 514–525, 2009.

[78] K. Harada, J. Sakuma, I. Ono, and S. Kobayashi, "Constraint-handling method for multi-objective function optimization: Pareto descent repair operator," in *EMO'07: Proc. of the 4th International Conference on Evolutionary Multi-Criterion Optimization*, 2006, pp. 156–170.

[79] L. Jiao, J. Luo, R. Shang, and F. Liu, "A modified objective function method with feasible-guiding strategy to solve constrained multi-objective optimization problems," *Appl. Soft Comput.*, vol. 14, pp. 363–380, 2014.

[80] T. A. Moebes, "Text data mining applied to clustering with cost effective tools," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Waikoloa, Hawaii, USA, October 10-12, 2005*, 2005, pp. 2787–2794. [Online]. Available: https://doi.org/10.1109/ICSMC.2005.1571572

[81] A. Osyczka and S. Kundu, "A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm," *Structural optimization*, vol. 10, no. 2, pp. 94–99, 1995.

[82] M. Tanaka, H. Watanabe, Y. Furukawa, and T. Tanino, "Ga-based decision support system for multicriteria optimization," in *Systems, Man and Cybernetics, 1995. Intelligent Systems for the 21st Century., IEEE International Conference on*, vol. 2. IEEE, 1995, pp. 1556–1561.

[83] N. Srinivas and K. Deb, "Muiltiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary computation*, vol. 2, no. 3, pp. 221–248, 1994.

[84] K. Deb, *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, 2001, vol. 16.

[85] Q. Zhang, A. Zhou, S. Zhao, P. N. Suganthan, W. Liu, and S. Tiwari, "Multiobjective optimization test instances for the cec 2009 special session and competition," *University of Essex, Colchester, UK and Nanyang technological University, Singapore, special session on performance assessment of multi-objective optimization algorithms, technical report*, vol. 264, 2008.

[86] V. Aragon, S. Esquivel, and C. A. C. Coello, "Evolutionary multiobjective optimization in non-stationary environments," *Journal of Computer Science and Technology*, vol. 5, no. 3, pp. 133–143, 2005.

[87] C. R. B. Azevedo and A. F. R. Araújo, "Generalized immigration schemes for dynamic evolutionary multiobjective optimization," in *CEC'11: Proc. of the 2011 IEEE Congress on Evolutionary Computation*, 2011, pp. 2033–2040.

[88] I. Hatzakis and D. Wallace, "Dynamic multi-objective optimization with evolutionary algorithms: a forward-looking approach," in *Genetic and Evolutionary Computation Conference, GECCO 2006, Proceedings, Seattle, Washington, USA, July 8-12, 2006*, 2006, pp. 1201–1208.

[89] ——, "Topology of anticipatory populations for evolutionary dynamic multiobjective optimization," in *AIAA/ISSMO'06: Proc. of the 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2006, pp. 1–10.

[90] Q. Zhang, A. Zhou, and Y. Jin, "RM-MEDA: A regularity model-based multiobjective estimation of distribution algorithm," *IEEE Trans. Evolutionary Computation*, vol. 12, no. 1, pp. 41–63, 2008.

[91] A. Zhou, Y. Jin, and Q. Zhang, "A population prediction strategy for evolutionary dynamic multiobjective optimization," *IEEE Trans. Cybernetics*, vol. 44, no. 1, pp. 40–53, 2014.

[92] A. Zhou, Y. Jin, Q. Zhang, B. Sendhoff, and E. P. K. Tsang, "Prediction-based population re-initialization for evolutionary dynamic multi-objective optimization," in *Evolutionary Multi-Criterion Optimization, 4th International Conference, EMO 2007, Matsushima, Japan, March 5-8, 2007, Proceedings*, 2006, pp. 832–846.

[93] Z. Peng, J. Zheng, J. Zou, and M. Liu, "Novel prediction and memory strategies for dynamic multiobjective optimization," *Soft Comput.*, vol. 19, no. 9, pp. 2633–2653, 2015.

[94] Y. Wu, Y. Jin, and X. Liu, "A directed search strategy for evolutionary dynamic multiobjective optimization," *Soft Comput.*, vol. 19, no. 11, pp. 3221–3235, 2015.

[95] W. T. Koo, C. K. Goh, and K. C. Tan, "A predictive gradient strategy for multi-objective evolutionary algorithms in a fast changing environment," *Memetic Computing*, vol. 2, no. 2, pp. 87–110, 2010.

[96] A. Muruganantham, K. C. Tan, and P. Vadakkepat, "Evolutionary dynamic multi-objective optimization via kalman filter prediction," *IEEE Trans. Cybernetics*, 2015, accepted for publication.

[97] Y. Wang and B. Li, "Investigation of memory-based multi-objective optimization evolutionary algorithm in dynamic environment," in *CEC'09: Proc. of the 2009 IEEE Congress on Evolutionary Computation*, 2009, pp. 630–637.

[98] Z. Peng, J. Zheng, and J. Zou, "A population diversity maintaining strategy based on dynamic environment evolutionary model for dynamic multiobjective optimization," in *CEC'14: Proc. of the 2014 IEEE Congress on Evolutionary Computation*, 2014, pp. 274–281.

[99] B. Zheng, "A new dynamic multi-objective optimization evolutionary algorithm," in *ICNC'07: Proc. of the 2007 International Conference on Computing, Networking and Communications*, 2007, pp. 565–570.

[100] R. Liu, W. Zhang, L. Jiao, F. Liu, and J. Ma, "A sphere-dominance based preference immune-inspired algorithm for dynamic multi-objective optimization," in *GECCO'10: Proc. of the 2010 Genetic and Evolutionary Computation Conference*, 2010, pp. 423–430.

[101] C. K. Goh and K. C. Tan, "A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization," *IEEE Trans. Evolutionary Computation*, vol. 13, no. 1, pp. 103–127, 2009.

[102] A. A. Montaño, C. A. C. Coello, and E. Mezura-Montes, "Multiobjective evolutionary algorithms in aeronautical and aerospace engineering," *IEEE Trans. Evolutionary Computation*, vol. 16, no. 5, pp. 662–694, 2012.

[103] A. Ponsich, A. L. Jaimes, and C. A. C. Coello, "A survey on multiobjective evolutionary algorithms for the solution of the portfolio optimization problem and other finance and economics applications," *IEEE Trans. Evolutionary Computation*, vol. 17, no. 3, pp. 321–344, 2013.

[104] H. R. Cheshmehgaz, M. N. Islam, and M. I. Desa, "A polar-based guided multi-objective evolutionary algorithm to search for optimal solutions interested by decision-makers in a logistics network design problem," *J. Intelligent Manufacturing*, vol. 25, no. 4, pp. 699–726, 2014.

[105] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "Automatic programming via iterated local search for dynamic job shop scheduling," *IEEE Trans. Cybernetics*, vol. 45, no. 1, pp. 1–14, 2015.

[106] J. Branke, S. Nguyen, C. Pickardt, and M. Zhang, "Automated design of production scheduling heuristics: A review," *IEEE Trans. Evolutionary Computation*, vol. 20, no. 1, pp. 110–124, 2016.

[107] S. U. Guan, Q. Chen, and W. Mo, "Evolving dynamic multi-objective optimization problems with objective replacement," *Artif. Intell. Rev.*, vol. 23, no. 3, pp. 267–293, 2005.

[108] R. Azzouz, S. Bechikh, and L. B. Said, "A multiple reference point-based evolutionary algorithm for dynamic multi-objective optimization with undetectable changes," in *CEC'14: Proc. of the 2014 IEEE Congress on Evolutionary Computation*, 2014, pp. 3168–3175.

[109] H. Ishibuchi, N. Akedo, and Y. Nojima, "Behavior of multiobjective evolutionary algorithms on many-objective knapsack problems," *IEEE Trans. Evolutionary Computation*, vol. 19, no. 2, pp. 264–283, 2015.

[110] B. Li, J. Li, K. Tang, and X. Yao, "Many-objective evolutionary algorithms: A survey," *ACM Comput. Surv.*, vol. 48, no. 1, p. 13, 2015.

[111] K. V. Price, "Differential evolution," in *Handbook of Optimization - From Classical to Modern Approach*, ser. Intelligent Systems Reference Library. Springer, 2013, vol. 38, pp. 187–214.

[112] K. Praditwong and X. Yao, "A new multi-objective evolutionary optimisation algorithm: The two-archive algorithm," in *CIS'06: Proc. of the 2006 International Conference Computational Intelligence and Security*, 2006, pp. 95–104.

[113] B. Li, J. Li, K. Tang, and X. Yao, "An improved two archive algorithm for many-objective optimization," in *CEC'14: Proc. of the 2014 IEEE Congress on Evolutionary Computation*, 2014, pp. 2869–2876.

[114] M. D. McKay, R. J. Beckman, and W. J. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, no. 2, pp. 239–245, 1979.

[115] K. Deb and M. Goyal, "A combined genetic adaptive search (GeneAS) for engineering design," *Computer Science and Informatics*, vol. 26, pp. 30–45, 1996.

[116] M. Farina, K. Deb, and P. Amato, "Dynamic multiobjective optimization problems: test cases, approximations, and applications," *IEEE Trans. Evolutionary Computation*, vol. 8, no. 5, pp. 425–442, 2004.

[117] M. Helbig and A. P. Engelbrecht, "Benchmarks for dynamic multi-objective optimisation algorithms," *ACM Comput. Surv.*, vol. 46, no. 3, pp. 37:1–37:39, 2014.

[118] S. Jiang and S. Yang, "Evolutionary dynamic multi-objective optimization: benchmarks and algorithm comparisons," *IEEE Trans. Cybernetics*, 2016, accepted for publication.

[119] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[120] V. R. Khare, X. Yao, and K. Deb, "Performance scaling of multi-objective evolutionary algorithms," in *EMO'03: Proc. of the 2nd International Conference on Evolutionary Multi-Criterion Optimization*, 2003, pp. 376–390.

[121] R. Shang, L. Jiao, Y. Ren, J. Wang, and Y. Li, "Immune clonal coevolutionary algorithm for dynamic multiobjective optimization," *Natural Computing*, vol. 13, no. 3, pp. 421–445, 2014.

[122] S. Jiang and S. Yang, "A steady-state and generational evolutionary algorithm for dynamic multiobjective optimization," *IEEE Trans. Evolutionary Computation*, 2016, accepted for publication.

[123] E. J. Hughes, "Evolutionary many-objective optimisation: many once or one many?" in *CEC'05: Proc. of the 2005 IEEE Congress on Evolutionary Computation*, 2005, pp. 222–227.

[124] P. Kruchten, *The Rational Unified Process - An Introduction, 3rd Edition*, ser. Addison Wesley object technology series. Addison-Wesley, 2004. [Online]. Available: http://vig.pearsoned.com/store/product/1,1207,store-12521_isbn-0321197704,00.html

[125] R. Farmani, G. A. Walters, and D. A. Savic, "Trade-off between total cost and reliability for anytown water distribution network," *Journal of Water Resources Planning and Management*, vol. 131, no. 3, pp. 161–171, 2005.

[126] T. K. Abdel-Hamid and S. E. Madnick, "The dynamics of software project scheduling," *Communications of the ACM*, vol. 26, no. 5, pp. 340–346, 1983.

[127] F. Padberg, "A discrete simulation model for assessing software project scheduling policies," *Software Process: Improvement and Practice*, vol. 7, no. 3-4, pp. 127–139, 2002.

[128] B. Boehm, B. Clark, E. Horowitz, C. Westland, R. Madachy, and R. Selby, "Cost models for future software life cycle processes: Cocomo 2.0," *Annals of software engineering*, vol. 1, no. 1, pp. 57–94, 1995.

[129] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex Systems*, vol. 9, 1994.

[130] K. Deb and M. Goyal, "A combined genetic adaptive search (GeneAS) for engineering design," *Computer Science and Informatics*, vol. 26, pp. 30–45, 1996.

[131] H. Mala-Jetmarova, N. Sultanova, and D. A. Savic, "Lost in optimisation of water distribution systems? A literature review of system operation," *Environmental Modelling and Software*, vol. 93, pp. 209–254, 2017.

[132] T. M. Walski, "The wrong paradigm–why water distribution optimization doesn't work," *Journal of Water Resources Planning and Management*, vol. 127, no. 4, pp. 203–205, 2001.

[133] http://emps.exeter.ac.uk/engineering/research/cws/resources/benchmarks/expansion/anytown.php.

[134] E. Todini, "Looped water distribution networks design using a resilience index based heuristic approach," *Urban Water*, vol. 2, no. 2, pp. 115–122, 2000.

[135] S. Saleh and T. Tanyimboh, "Multi-directional maximum-entropy approach to the evolutionary design optimization of water distribution systems," *Water Resources Management*, vol. 30, no. 6, pp. 1885–1901, 2016.

[136] R. Farmani, G. Walters, and D. Savic, "Evolutionary multi-objective optimization of the design and operation of water distribution network: total cost vs. reliability vs. water quality," *Journal of Hydroinformatics*, vol. 8, no. 3, pp. 165–179, 2006.

[137] https://www.epa.gov/water-research/epanet#downloads.

[138] G. Fu, Z. Kapelan, J. R. Kasprzyk, and P. Reed, "Optimal design of water distribution systems using many-objective visual analytics," *Journal of Water Resources Planning and Management*, vol. 139, no. 6, pp. 624–633, 2013.